Faster Than Weighted A*:
An Optimistic Approach to Bounded Suboptimal Search

Jordan T. Thayer and **Wheeler Ruml**
Department of Computer Science
University of New Hampshire
Durham, NH 03824 USA
`jtd7`, `ruml` at `cs.unh.edu`

Abstract

Planning, scheduling, and other applications of heuristic search often demand we tackle problems that are too large to solve optimally. In this paper, we address the problem of solving shortest-path problems as quickly as possible while guaranteeing that solution costs are bounded within a specified factor of optimal. 38 years after its publication, weighted A* remains the best-performing algorithm for general-purpose bounded suboptimal search. However, it typically returns solutions that are better than a given bound requires. We show how to take advantage of this behavior to speed up search while retaining bounded suboptimality. We present an optimistic algorithm that uses a weight higher than the user's bound and then attempts to prove that the resulting solution adheres to the bound. While simple, we demonstrate that this algorithm consistently surpasses weighted A* in four different benchmark domains including temporal planning and gridworld pathfinding.

Introduction

Many difficult problems, including planning, can be represented as shortest-path problems. If sufficient resources are available, optimal solutions to these problems can be found using A* search with an admissible heuristic (Hart, Nilsson, and Raphael 1968). However, in many practical scenarios, one is willing to accept a suboptimal solution in return for reduced computation time. In this paper, we will consider the setting in which one wants the fastest search possible while guaranteeing that the suboptimality of the resulting solution is bounded to within a given factor of the optimal solution's cost. For a given factor w, we say that an algorithm is w-admissible.

The best previously proposed algorithm for this problem is weighted A* (Pohl 1970), in which the traditional node evaluation function f is modified to place additional weight on the heuristic evaluation function h, as in $f'(n) = g(n) + w \cdot h(n)$, with $w \geq 1$. By penalizing nodes with large h values, the search becomes greedier, which often results in finding a solution faster. The solution returned by weighted A* is w-admissible. Weighted A* is beautifully simple and often performs well, but other algorithms have been proposed. One is dynamically weighted A* (Pohl 1973), which

requires an estimate of the depth of the solution and then decreases w from its original value at the root of the tree to 1 at the estimated goal depth. This maintains w-admissibility. Another algorithm is A_ϵ^* (Pearl and Kim 1982), which requires both the traditional estimate of cost-to-go h and also an estimate of the search effort or search distance-to-go d. Of all the nodes in the open list with an f value within a factor of w of the minimum f value of any node in open, A_ϵ^* expands that node whose d value is minimum. A_ϵ^* is also w-admissible. These two newer algorithms have not displaced weighted A*, which remains widely used, and in the experiments reported below we will see that they do not find solutions as quickly.

In this paper, we propose a new technique for fast w-admissible search, called optimistic search. The algorithm has two phases. First, it employs an aggressively greedy search phase to find a solution that is not guaranteed to be w-admissible, but usually is. This is followed by a 'clean-up' phase, in which the algorithm attempts to prove that the solution obtained by the aggressive part of the search is w-admissible. After showing the theoretical intuition behind this optimistic approach, we demonstrate empirically that it performs well across a variety of search problems. Given its simplicity, we believe optimistic search can find wide use in AI systems.

An Optimistic Approach

We begin by noting that previous approaches to suboptimal heuristic search, including weighted A*, dynamically weighted A*, and A_ϵ^*, are very strict. That is, no node is ever expanded which could not lead to a w-admissible goal. Consider weighted A*, for example. Its w-admissibility derives from the following straightforward reasoning, which we will build up in stages for later reuse. We will assume that h is admissible. The optimal cost of a path from the root to a node n will be notated $g^*(n)$ and *opt* will represent the node at the end of an optimal path to a solution. We start with a special node p:

Lemma 1 (following Pearl, 1984) *Let p be the deepest node on* open *that lies along the optimal path to* opt. *No matter how a best-first search selects nodes for expansion, $f(p) = g(p) + h(p) \leq g^*(\text{opt})$.*
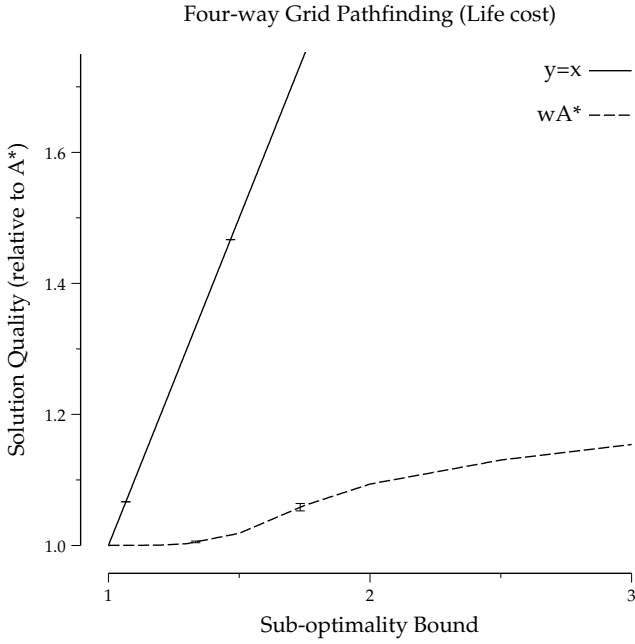
Figure 1: The suboptimality of solutions returned by weighted A* versus the suboptimality bound, averaged over 20 grid-world path-finding problems.

Optimistic Search(*initial*, *bound*)
1. $open_f \leftarrow \{initial\}$
2. $open_{\widehat{f}} \leftarrow \{initial\}$
3. $incumbent \leftarrow \infty$
4. repeat until $bound \cdot f(\text{first on } open_f) \geq f(incumbent)$:
5.   if $\widehat{f}(\text{first on } open_{\widehat{f}}) < \widehat{f}(incumbent)$ then
6. $n \leftarrow$ remove first on $open_{\widehat{f}}$
7.   remove n from $open_f$
8.   else $n \leftarrow$ remove first on $open_f$
9.   remove n from $open_{\widehat{f}}$
10.   add n to *closed*
11.   if n is a goal then
12. $incumbent \leftarrow n$
13.   else for each child c of n
14.   if c is duplicated in $open_f$ then
15.   if c is better than the duplicate then
16.   replace copies in $open_f$ and $open_{\widehat{f}}$
17.   else if c is duplicated in *closed* then
18.   if c is better than the duplicate then
19.   add c to $open_f$ and $open_{\widehat{f}}$
20.   else add c to $open_f$ and $open_{\widehat{f}}$

Figure 2: Optimistic search using an admissible node evaluation function f and an inadmissible function $\widehat{f}$.

Proof: Let p be the deepest node on *open* that lies along an optimal path to *opt*. Such a node must exist because an optimal path to *opt* exists by definition. The root is on it, and if a node which has been expanded is on it, one of the children must be, and all children are inserted into *open*. $f(p) \leq f(opt) = g^*(opt)$ by our definition of p and the admissibility of h. $\square$

This wonderful property of p supports a general result for weighted A*:

Theorem 1 (after Pohl, 1970) *For any node n expanded by a best-first search guided by $f' = g(n) + w \cdot h(n)$, $f'(n) \leq w \cdot g^*(\text{opt})$.*

Proof: Consider an optimal path to *opt*. If all nodes on this path have been expanded, we have the optimal solution and the theorem holds trivially. Otherwise, let p be the deepest node on *open* that lies along the optimal path to *opt*. When we expand n, $f'(n) \leq f'(p)$ because n was selected for expansion before p. $f'(p) = g(p) + w \cdot h(p) \leq w \cdot (g(p) + h(p)) = w \cdot f(p)$ by algebra. So we have $f'(n) \leq w \cdot f(p)$. By Lemma 1, $w \cdot f(p) \leq w \cdot f(opt) = w \cdot g^*(opt)$. $\square$

The w-admissibility of weighted A* is just a special case:

Corollary 1 *For the solution s returned by weighted A*, $g(s) \leq w \cdot g^*(\text{opt})$.*

Proof: Because s is a goal node and h is admissible, $h(s) = 0$. So $g(n) = f(n) = f'(n)$ by the definition of f' and $f'(n) \leq w \cdot g^*(opt)$ by Theorem 1. $\square$

While this strict approach to w-admissibility has a certain conservative appeal, there is also a long history in AI research of exploiting the fact that worst case behavior rarely occurs. This leads us to an optimistic approach in which we allow ourselves to expand nodes more aggressively, without a strict guarantee that they lead to a w-inadmissible solution. Any fast search method with an open list can be used—in the experiments below, we use weighted A* with a high weight. Crucially, once we have found a solution using the aggressive search, we will then expand additional nodes until we can either prove our solution is w-admissible or we find a better one. This proof of w-admissibility relies on the following corollary of Lemma 1:

Corollary 2 (following Pearl, 1984, and Hansen and Zhou, 2007) *No matter how an open list-based search algorithm selects nodes for expansion, the lowest f value of any node on the open list is a lower bound on the optimal solution cost.*

Proof: Consider node p in Lemma 1. The lowest f on open will be $\leq f(p)$. $\square$

This means that, to prove that a solution found by an optimistic search is w-admissible, we can expand the node in open with the lowest f value until the lowest f value in open is within a factor of w of the solution's cost.

The clear risk in such a technique is that the solution found during the first phase might not be w-admissible. This will cause the algorithm to behave like A*, expanding all nodes with f values less than the optimal solution after having already invested time into producing a suboptimal solution. We attempt to hedge against this worst case: if there is a node whose inadmissible heuristic value according to the aggressive search is less than that of the incumbent solution, then that node is selected for expansion. Figure 2 gives a complete sketch of the optimistic search approach. In an op-

timized implementation, one would probably delay forming the $open_f$ list until the first solution was found. Note that the aggressive heuristic $\widehat{f}$ can be any arbitrarily inadmissible function.

Why would we expect this technique to work? It depends heavily on the aggressive search finding a node which falls within the desired suboptimality bound. When using weighted A* with a weight higher than w as the aggressive search component, the proof of Theorem 1 shows us how this can happen. Recall the crucial step:

$$f'(p) = g(p) + w \cdot h(p) \leq w \cdot (g(p) + h(p)) = w \cdot f(p)$$

Note the significant potential gap between $g(p) + w \cdot h(p)$ and $w \cdot (g(p) + h(p))$. Equality only holds when $g(p) = 0$. In many heuristic search problems, this happens only at the root node. Everywhere else, there is a significant gap, which implies that nodes whose f' is less than $f'(p)$ are usually significantly better than $w \cdot g^*(opt)$. This is exactly the gap that optimistic search is trying to exploit.

The gap present in the inequality also manifests in empirical examinations of weighted A*. Figure 1 shows the true quality of solutions found by weighted A* compared to the suboptimality bound, and used weighting factor, w. The line $y = x$ is also included to show the theoretical limit. Clearly, not only is it possible for weighted A* to return solutions far better than the bound suggests, it is also common in some domains.

A higher weight in weighted A* usually results in faster search and the decrease in solution quality is often not linear, as the bound would suggest. By running an aggressive search first with a weight higher than the bound, and then proving our solution is within the bound, we can expect faster total search time for a w-admissible solution than when running weighted A* itself.

Empirical Evaluation

To gain a more concrete sense of the behavior of optimistic search, we implemented it and several other search algorithms and tested them on four challenging benchmark search problems: temporal planning, grid-world pathfinding, the traveling salesman problem, and the sliding tile puzzle. All algorithms were implemented in Objective Caml, compiled to 64-bit native code executables, and run on a collection of Intel Linux systems. We implemented the following algorithms:

weighted A* (wA*) using the desired suboptimality bound as a weight. For domains with significant number of duplicates, we also implemented a version of weighted A* that ignores nodes that are already in the closed list, noted by dd in our graphs. Likhachev, Gordon, and Thrun (2004) point out that this modification retains w-admissibility while potentially reducing the number of nodes expanded (although see Hansen and Zhou, 2007, for another view). Such an approach is only possible with a consistent heuristic.

dynamically weighted A* using $d(root)$ as a depth bound.

A^*_ϵ using the desired sub-optimality bound to form the 'focal' list from which the node to expand is selected

optimistic search using weighted A* with a weight of $2(bound-1)+1$ for the aggressive search phase. This was chosen arbitrarily—a different weight may have given better results.

bounded anytime weighted A* using a weight of $2(bound-1)+1$. Bounded anytime weighted A* (BAwA*) is a natural extension of the anytime weighted A* algorithm introduced by Hansen and Zhou (2007). Standard implementations of this algorithm would run until they converged on the optimal solution. By adding an additional priority queue sorted on f, we can keep track of the node with the lowest f value. With the addition of the second priority queue, the algorithm can be made to run until its incumbent solution can be shown to be within the desired suboptimality bound. This is functionally equivalent to our optimistic search without a cleanup phase. Again, the weight here was arbitrarily chosen to allow for a direct comparison to optimistic search, a different weight may have resulted in better performance.

Our primary figure of merit is the speed with which an algorithm can find a solution that is guaranteed to fall within a given suboptimality bound.

Temporal Planning

Heuristic search algorithms have been widely applied to planning problems (Bonet and Geffner 2001; Zhou and Hansen 2006). It is a domain in which optimal solutions can be extremely expensive to obtain (Helmert and Röger 2007). We tested our algorithms on 31 temporal planning problems from five benchmark domains taken from the 1998 and 2002 International Planning Competitions where the objective function is to minimize the plan duration (makespan).

To find the plan, we used the temporal regression planning framework in which the planner searches backwards from the goal state S_G to reach the initial state S_I (Bonet and Geffner 2001). To guide the search, we compute $h(n)$ using the admissible $\overline{H}^2$ heuristic of the TP4 planner (Haslum and Geffner 2001). This heuristic estimates the shortest makespan within which each single predicate or pair of predicates can be reached from the initial state S_I. This is computed once via dynamic programming before starting the search, taking into account the pairwise mutual exclusion relations between actions in the planning problem. In order to compute a search-distance-to-go function d, we also computed the expected number of steps to reach the shortest makespan solution. This value was estimated by first extracting a relaxed plan (Hoffmann and Nebel 2001) that approximates the closest shortest solution in terms of makespan from a given search node. The number of regression steps in this plan is then used as the distance estimate to the cheapest solution.

Figure 3 shows results on the hardest benchmark problem from each domain that A* could solve within four minutes. The x axis represents the sub-optimality bound, where 1 is optimal and 3 is three times the optimal cost. Samples were taken at 1, 1.001, 1.005, 1.01, 1.05, 1.1, 1.15, 1.2, 1.3, 1.5, 1.75, 2, 2.52 and 3. The y axis is the number of nodes gen-
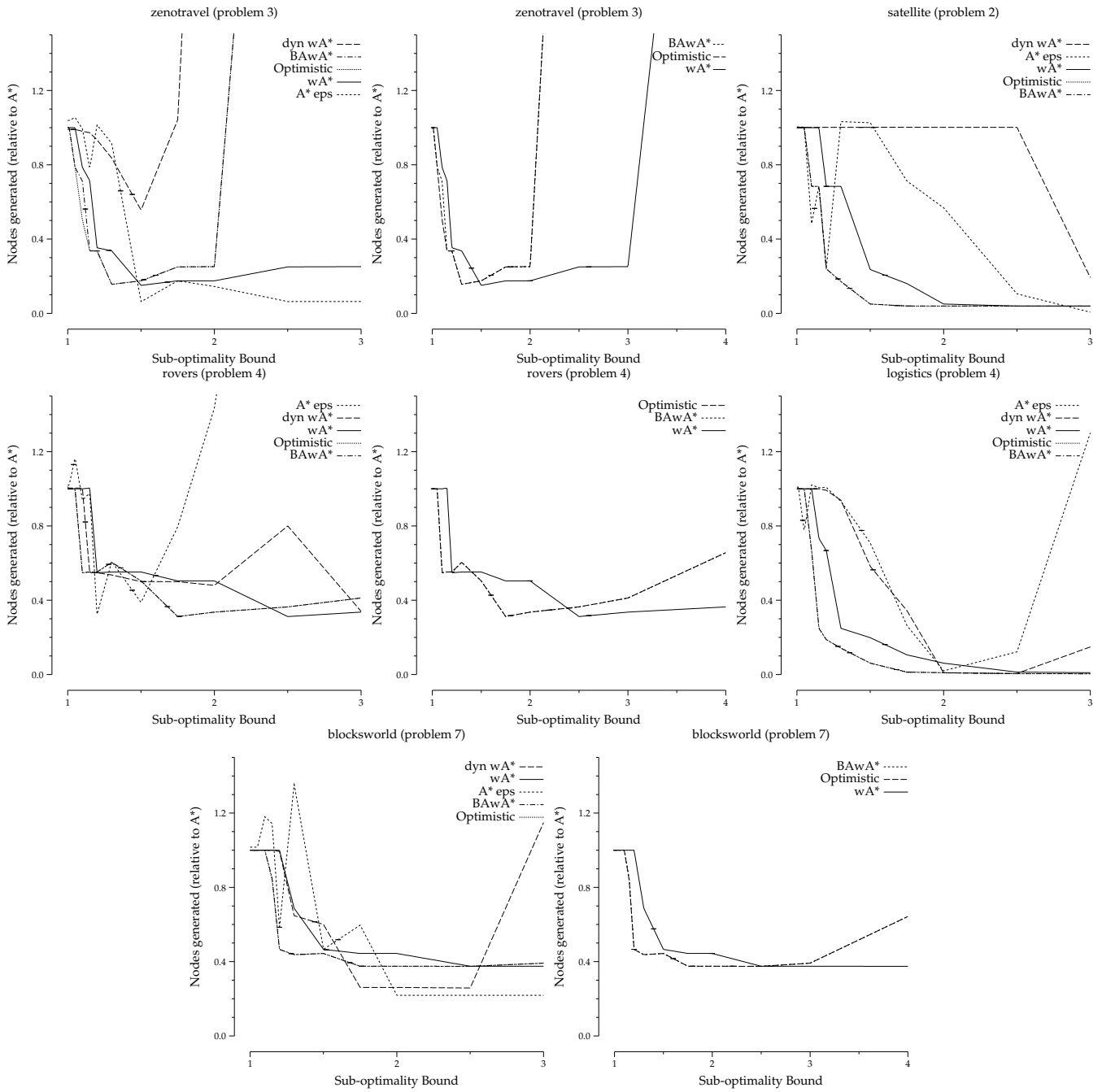
Figure 3: Performance on difficult temporal planning problems.

erated, normalized by the number of nodes generated by A* search on the same problem. Zenotravel, rovers, and blocks world are presented twice in order to provide a clearer picture of the relationship between weighted A*, bounded anytime weight A*, and optimistic search.

Optimistic search performed as expected in this domain, just as weighted A* would have done with a higher weight. In effect, it shifts the curve for weighted A* to the left. For small suboptimality bounds, this gives rise to large speed-

ups, often reducing the search time by 50%. In zenotravel, where weighted A* degenerates with a high weight, optimistic search shifts the curve to the left and degenerates faster. Clearly, if one knows the performance of weighted A*, it is easy to predict the performance of optimistic search and either achieve the same suboptimality bound within fewer node generations or achieve a better suboptimality bound for the same number of node generations. This extends to any inadmissible search which could be used for

the aggressive phase of our algorithm. If the performance of what will be the aggressive search is known a priori, tuning the performance of optimistic search is as simple as selecting the right level of optimism for the first phase of the search.

Bounded anytime weighted A* behaves very much like optimistic search, in many cases they behave identically. In satellite, logistics, rovers, and blocksworld, the lines of optimistic search and bounded anytime weighted A* lie on top of one another. This should be expected since both are performing the same search initially, and differ in their approach to confirming that the solution is within the given suboptimality bound. In planning, finding the initial solution dominates the cleanup phase, wherein the initial solutions are shown to be w-admissible. Given the importance of an explicit cleanup phase in other domains, it is surprising how often the two approaches perform identically here.

In cases where optimistic search performs better, it is because its explicit cleanup strategy allows it to prove the quality of the solution faster than bounded anytime weighted A* can. This shouldn't be surprising, as it says that, typically, expanding nodes in f order raises the lower bound on solution quality faster than expanding nodes in f' order would. The lower bound on solution quality is the smallest f in our open list, and so clearly expanding nodes in f order will raise this value fastest.

There are several reasons for which the algorithms might perform identically. The first, and most common, is that no cleanup phase is needed. If, during the initial aggressive search, all nodes whose f values were within a factor of the suboptimality bound of the aggressive solution were expanded, no clean up is needed. This can happen when the suboptimality bound is very generous because the suboptimality guarantee allows the solution to differ greatly from the lower bound. Alternatively, when the suboptimality bound is incredibly tight, we aren't likely to have need of a cleanup phase. In this case, optimistic search will have already expanded most, if not all, of the nodes whose f values are near that of the aggressive solution. Finally, the algorithms perform similarly when h is extremely accurate, as this also eliminates the cleanup phase.

Dynamically weighted A* usually performs much worse than plain weighted A*, with the exception of blocksworld, where it temporarily surpasses weighted A* and optimistic search. A^*_ϵ can perform well in planning, as particularly evident in zenotravel, but its behavior is erratic in many of the temporal planning problems, and it performs very poorly in other domains. In general, if it performs well, it performs well when the suboptimality bound is very high, allowing A^*_ϵ to function primarily as if it were a greedy search on $d(n)$.

Grid-world Planning

We considered 12 varieties of simple path planning problems on a 2000 by 1200 grid, using either 4-way or 8-way movement, three different probabilities of blocked cells, and two different cost functions. The start state was in the lower left corner and the goal state was in the lower right corner. In addition to the standard unit cost function, under which moves have the same cost everywhere, we tested a version the uses a graduated cost function in which moves along the

upper row are free and the cost goes up by one for each lower row. We call this cost function 'life' because it shares with everyday living the property that a short direct solution that can be found quickly (shallow in the search tree) is relatively expensive while a least-cost solution plan involves many annoying economizing steps. In 8-way movement worlds, diagonal movement costs $\sqrt{2}$ times as much as movement in any of the cardinal directions. Under both cost functions, simple analytical lower bounds (ignoring obstacles) are available for the cost $g(n)$ and distance $d(n)$ (in search steps) to the cheapest goal. The obstacle density introduces error to the heuristics and challenge to the problems.

Figure 4 shows the algorithms' performances on the hardest problems we considered in each of the four classes of worlds (35% blocked cells in the four-way worlds, 45% in the eight-way worlds). The x axis represents the suboptimality bound used, with 1 being optimal and 3 being three times the optimal solution cost. Samples were taken at the same points as in temporal planning. The y axis is the number of generated nodes relative to an optimal A* search, averaged over 20 random worlds. Error bars indicate 95% confidence intervals around the mean, although they are typically so tight as to be invisible. All algorithms are shown in the leftmost plot in each row, with the next two plots showing fewer algorithms in greater detail.

Again we see optimistic search perform as if it were weighted A* running with a higher weight. This is clearest in the Life four-way graph. Bounded anytime weighted A* is not nearly as competitive here as it was in temporal planning. Although both algorithms take the same amount of time in finding their first answer, bounded anytime weighted A* spends more time proving the quality of the solution than optimistic search. The benefit of expanding on f instead of f', so long as the incumbent solution is within the desired suboptimality bound, is clear.

Dynamically weighted A* and A^*_ϵ perform quite poorly, running off the top of all the plots. In the unit-cost problems, optimistic search is able to boost weighted A*'s performance enough to match the special duplicate dropping version (notated 'wA* dd' in the plots), actually surpassing it on the eight-way problems (the duplicate dropping version is almost identical to the plain weighted A* line). In life-cost problems, duplicate dropping seems essential. For all but the largest weights, where A^*_ϵ becomes quite competitive, weighted A* with duplicate dropping outperforms all of the other algorithms. When comparing optimistic search with weighted A*, we do see the same pattern of performance as before: optimistic search provides the performance of weighted A* running at a higher bound, while returning solutions within a lower bound. This can be desirable, as it is in unit cost worlds, or it can be detrimental, as it is early on in life cost worlds.

Traveling Salesman

Following Pearl and Kim (1982) we also tested on a straightforward encoding of the traveling salesman problem. Each node represents a partial tour with children representing the choice of which city to visit next. We used the minimum
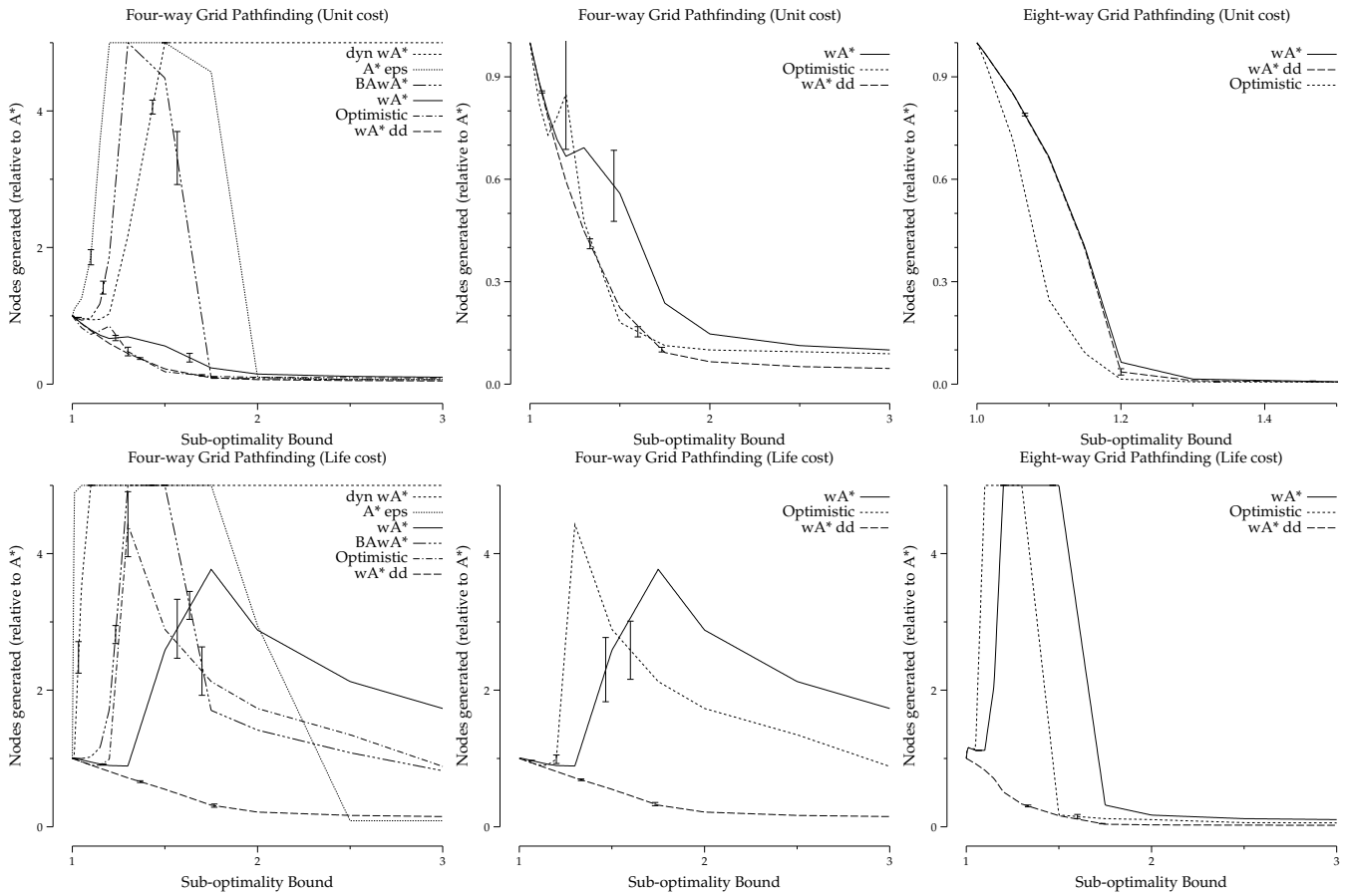
Figure 4: Performance on grid-world path-finding problems.

spanning tree heuristic for $h(n)$ and the exact depth remaining in the tree for $d(n)$. Again, samples were taken at the same points as in temporal planning.

Figure 5 shows the algorithms' performance on two types of problems: 19 cities placed uniformly in a unit square ('usquare') and 12 cities with distance chosen uniformly at random between 0.75 and 1.25 ('Pearl and Kim Hard'). Both types of problems were symmetric, and results are averages over 40 instances. In both types of problems, the results are clear: optimistic search improves over weighted A* and dynamically weighted A* lags behind. Bounded anytime weighted A* performs better than either A_ϵ^* or dynamically weighted A*, and at some points it is even better than weighted A*, but the focus optimistic search demonstrates during its cleanup phase proves to be very useful in obtaining w-admissible solutions quickly.

Branch and bound, a typical, and remarkably effective, approach for solving traveling salesman problems is absent from the results presented here. While it works very well for problems of fixed depth with few or no duplicate states, domains such as traveling salesman, branch and bound struggles in domains with duplicates, such as tiles and grid world, as well as in domains with infinite search spaces, such as temporal planning.

Sliding Tile Puzzles

Our last test domain is the venerable sliding tile puzzle. We tested on the 100 benchmark 15-puzzle instances from Korf (1985) using the standard Manhattan distance heuristic. Because A* cannot solve these instances within a reasonable memory bound, we normalize the number of nodes generated against iterative deepening A*. Several of the algorithms we tested ran into the same memory limitation as A*, so we show results only for weights of 1.075 and above.

Figure 6 shows the results. In the upper image we show the performance of all the algorithms, while below we show how the most competitive algorithms performed in this domain. Once again, optimistic search appears to be outperforming the other algorithms, and once again it appears to behave as if it were weighted A*, run with a higher weight. Although we can not provide measurements close to the optimal solution, we speculate that the algorithms behave much as they did in Grid-world. These two domains share a common trait in that they have many paths for reaching identical states. It is curious that dropping these duplicate states does not improve the performance of wA^* with duplicate dropping. While this might be a property of the problem, it seems more likely that it is merely a result of the higher suboptimality bounds being used here. A similar effect is present in grid-world, thought at a higher weight.
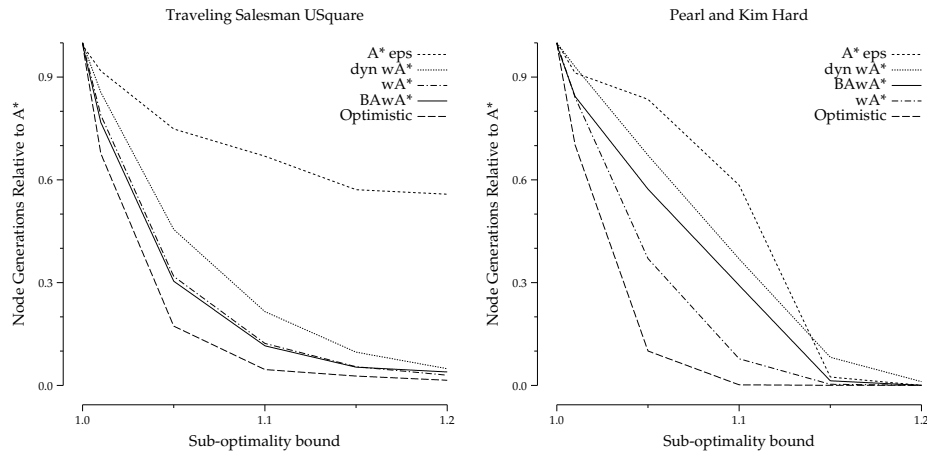
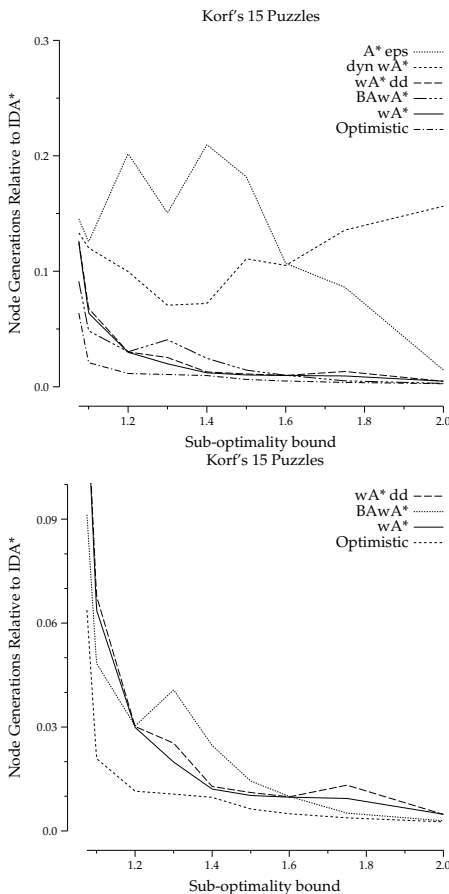Figure 5: Performance on traveling salesman problems.

Figure 6: Performance on 100 15-puzzle benchmark problems.

Discussion

When abandoning optimality, there are two basic strategies for retaining some control of the search: bound the time taken or bound the quality of the resulting solution. Anytime algorithms and real-time search are the main approaches used for finding solutions in a bounded time. For problems in which a solution with bounded sub-optimality is desirable, weighted A* has reigned for decades as the technique of choice. We have presented a new approach for using an inadmissible heuristic function in search and shown that it can deliver a solution within a desired sub-optimality bound faster than weighted A*.

Unlike previous techniques which expand only those nodes which are certain to lead to a w-admissible solution, optimistic search expands nodes which may not meet this strict criteria, which allows us to find solutions faster and widens the family of solutions available for search with bounded suboptimality. We have presented results using weighted A* as the aggressive search component; however, the only restriction on the aggressive component is that it must use an open list. The clean-up phase transforms an arbitrary inadmissible search into one that can provide bounded suboptimality. The expectation that the aggressive search will return a solution within the desired bound need only be reasonable, not absolute. Ideally, the method would be responsive to the provided bound, in that an increase in bound should allow for faster, if more costly, solutions. For example, if one were to use RTA* (Korf 1990), perhaps the depth of the look ahead should be proportional to the tightness of the suboptimality bound.

If one were solving many similar problem instances from the same domain, gathering data on the typical solution quality as a function of the search aggressiveness, as we saw displayed in Figure 1, might provide a basis for choosing the level of optimism that is appropriate for the desired bound. The $2(bound - 1) + 1$ formula we experimented with here is, judging by Figure 1, quite conservative. However, it already gives encouraging results. Presumably, a well tuned optimistic search would have even better performance.

Characterizing domains in which optimistic search performs poorly is difficult. A large part of the performance is dependant on the initial aggressive search. If this search performs poorly, the whole search will suffer. Even if the initial aggressive search returns a solution within the bound in a short amount of time, it is possible for optimistic search

to struggle with the cleanup phase. It must expand all nodes whose f value differs from the cost of the solution returned by the aggressive search by more than a factor of the desired suboptimality bound. If there are many such nodes, optimistic search will struggle to prove that the answer it has in hand is admissible.

Conclusions

We have addressed the problem of heuristic search with bounded suboptimality by introducing a new approach: optimistic search. In contrast to the strict approach of weighted A*, optimistic search couples an aggressively greedy search that risks expanding nodes outside the bound with a clean-up phase that proves the resulting solution does lie within the bound. In experiments across four different types of problems, we showed that this simple technique is predictable and effective, yielding results similar to those of weighted A* running with a looser bound. Guided by its proof of w-admissibility, we gained some intuition about why it should be expected to perform well. Given its simplicity, we believe optimistic search can find wide use in AI systems.

Acknowledgments

References

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1–2):5–33.

Hansen, E. A., and Zhou, R. 2007. Anytime heuristic search. *Journal of Artificial Intelligence Research* 28:267–297.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions of Systems Science and Cybernetics* SSC-4(2):100–107.

Haslum, P., and Geffner, H. 2001. Heuristic planning with time and resources. In *Proceedings of ECP-01*.

Helmert, M., and Röger, G. 2007. How good is almost perfect? In *Proceedings of the ICAPS-2007 Workshop on Heuristics for Domain-independent Planning: Progress, Ideas, Limitations, Challenges*.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Korf, R. E. 1985. Iterative-deepening-A*: An optimal admissible tree search. In *Proceedings of IJCAI-85*, 1034–1036.

Korf, R. E. 1990. Real-time heuristic search. *Artificial Intelligence* 42:189–211.

Likhachev, M.; Gordon, G.; and Thrun, S. 2004. ARA*: Anytime A* with provable bounds on sub-optimality. In *Proceedings of NIPS 16*.

Pearl, J., and Kim, J. H. 1982. Studies in semi-admissible heuristics. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-4(4):391–399.

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

Pohl, I. 1970. Heuristic search viewed as path finding in a graph. *Artificial Intelligence* 1:193–204.

Pohl, I. 1973. The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computation issues in heuristic problem solving. In *Proceedings of IJCAI-73*, 12–17.

Zhou, R., and Hansen, E. 2006. Breadth-first heuristic search. *Artificial Intelligence* 170(4–5):385–408.