Identify Conditions For Re-expanding Nodes or Not Re-expanding Nodes in Weighted A*

Mehjabin Rahman
ID: 501200106
Email:mehjabin.rahman@ryerson.ca
Toronto Metropolitan University

Abstract

Weighted heuristic search is particularly successful for search problems with close-to-optimal solutions and can frequently identify a close-to-optimal solution in a small fraction of the time. The use of Weighted A* raises the possibility of re-expansions. The simple logical assumption is that re-expanding nodes results in finding shorter solutions than not re-expanding nodes, at the cost of expanding more nodes. But, there are other scenarios where other tendencies occur. This work is to find the conditions of trade-off between the between re-expanding and not re-expanding nodes in the context of Weighted A* (WA*). We then experimentally demonstrate the different tendencies on grid based maps.

Introduction

The most famous, and most simple suboptimal search algorithm is Weighted A* (WA*) (Pohl 1970) – is the focus of this work. Similar to A* , WA* uses two data structures: OPEN and CLOSED. Initially, OPEN contains only the start state and CLOSED is empty. At every iteration, the algorithm chooses the state in OPEN that minimizes the cost function f(n) = g(n) + W · h(n), where g(n) is the cost of the lowest cost path found so far from the start state to n, h(n) is an admissible heuristic estimate of the cost of reaching the goal from n, and W is a parameter. The chosen state is expanded, generating its successors. The expanded state is moved to CLOSED and its successors are inserted to OPEN. WA* continues until a goal state is chosen for expansion or OPEN is empty.

Local Inconsistency

Some states have multiple paths from the start state to them, of different costs. It is well-known that A* with a consistent heuristic (and thus the f-cost is monotonically increasing) expands a state only after the lowest cost path to it was found. This is not the case for non-monotonic f-functions such as the one used by WA* .This is called local inconsistency. In this case, a state n may be generated with a smaller g-value after it has been expanded and inserted to CLOSED. At this point, it can be removed from CLOSED and put back in OPEN. Re-expanding is optional. For every closed state which is seen with a smaller g-value, a decision needs to be made whether to re-expand it or not. In the later case, the newly seen state is just discarded. In this work, we tested always re-expand and never re-expand policies in varying situations. Our main contributions to this work are,

- Implement the weighted A* algorithm to the path-finding domain.
- Analyze the results of the problem instances by using always re-expanding.
- Analyze the results of the problem instances by using never re-expanding.
- Analyze the results of the problem instances by using always re-expanding and never re-expanding by varying the value of the weight w.
- Analyze the results by using different heuristics (consistent and inconsistent).
- Implement anytime weighted A* algorithm with combine the idea of any time Repairing A*, to analyze the impact of using a combination of never re-expanding and always re-expanding.

Related Work

Historically, inconsistent heuristics have been generally avoided when searching for optimal solutions because of the cost of re-expanding closed nodes with A* and the belief that inconsistent heuristics are hard to concoct (Ariel Felner 2011). (Uzi Zahavi and Sturtevant 207) This paper has demonstrated that inconsistent heuristics are easy to create and that a large speedup can be achieved when using them with IDA* and BPMX. This represents a significant change to the conventional wisdom for heuristic search. In paper (Valenzano, Sturtevant, and Schaeffer 2014), they have presented a formal analysis of the impact that not re-expanding nodes can have on the quality of solutions returned by a best-first search. In particular, the inconsistency of a heuristic along an optimal solution path is shown to bound the suboptimality of the solutions found when not performing re-expansions. This bound, which is tight, was then used to demonstrate that not re-expanding nodes in an A* search that uses an admissible heuristic will have a quadratic effect on solution quality in the worst case.

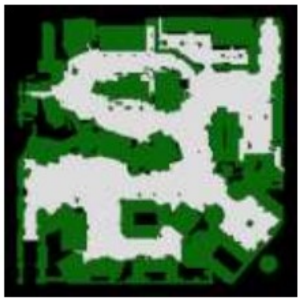
Figure 1: den400d
.

Description of the Work

Different possible relations between always re-expand and never re-expand can occur regarding the number of expanded nodes and the time needed to perform the search. We examine the following cases to identify the conditions for always re-expanding, never re-expanding, and a combination of both.

- The relations between always re-expanding and never re-expanding by only varying the weight for Weighted A* (WA*)
- The relations between always re-expanding and never re-expanding by only varying the heuristic function for Weighted A* (WA*). For this work, we have used the Manhattan distance, zero heuristic, and inconsistent heuristic. For inconsistent heuristics, we used the random selection procedure. We randomly choose between the Manhattan and zero heuristics to make it inconsistent.
- Then we analyze cases for Anytime wA*.
- And for the cases where Anytime wA* with no re-expansions will not converge to the optimal solution, we use the restarting policy. Restarting is often used in anytime algorithms for different purposes (Thayer and Ruml 2010). We first try to solve the problem by never re-expanding. During the execution of never re-expanding, we keep all nodes that were considered to be re-expand, in an inconsistency list. This list is returned to the OPEN after getting a solution. This is done repeatedly until we find an optimal solution. Then we examine the time needed and the number of nodes expansion for this combination.

Experimental results

This section presents experimental results for the Grid Path-Finding domain.

For our experiments, we chose two different maps from the Moving AI Lab. The maps that were chosen for this experiment, are shown in the following figures. We chose them since they have the characteristics of different domains.

The first map, figure 1, has a lot of wide open areas and no bottlenecks, while the last, figure2 has almost no open space and bottleneck density.

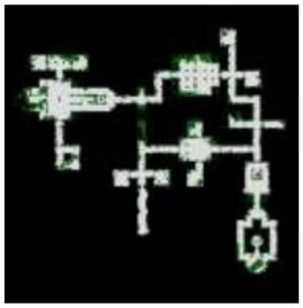
Figure 2: brc202d
.

Experimenting to find the necessity of re-expanding

It appears at first glance that increasing the number of re-expanding will result in more node expansions but improve the quality of the solutions. We tested wA* on 100 arbitrary examples both with and without re-expanding. Figure 3 shows that both policies often expand the same number of nodes. Additionally, in some cases, re-expanding expands fewer nodes than it would without it. Therefore, it is not entirely certain that never expanding will always lead to better solutions.

Experimenting with always re-expanding and never re-expanding policies: varying the weights (wA*)

We experimentally studied the different cases and connections between the quality of the found solution, measured by its suboptimality, and the number of expanded nodes. In figure 4 representative results are shown, for WA* with varying weights (w). We take values of w from 1.3 to 30. From figure 4 and table 1, we can see that with and without re-expanding, varying the weight, has almost no impact on the different cases (the yellow part illustrates this analysis).

w	without reexpand win	With reexpand win	Tie
1.3	16	18	66
1.5	15	19	66
2	17	20	63
2.5	8	20	72
3	11	27	62
5	10	20	70
10	10	16	74
20	11	14	75
30	9	13	18

Table 1: Experimental Result on map brc202d (wA*)

Experimenting with always re-expanding and never re-expanding policies: varying the heuristics

Figure 5, illustrates the number of nodes expands while using inconsistent heuristics. The shown result is for w=3. We can see that for most of the instances with re-expanding,

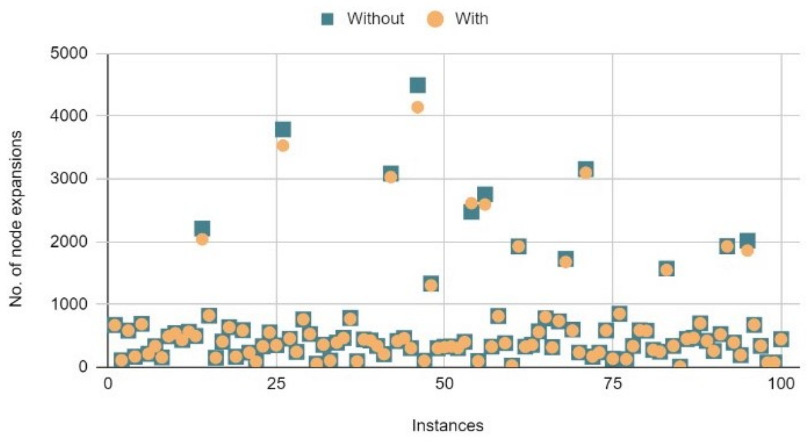

Figure 3: With or Without Re-expanding (wA*)

.

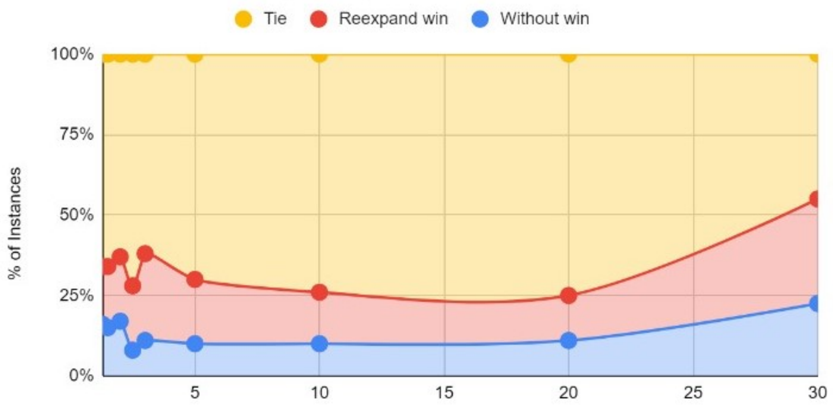

Figure 4: Never re-expanding win vs always expanding win vs tie

.

wA* expands more nodes. There are cases where the re-expanding policy expands to fewer nodes. In figure 5, instances 1, 5, 12, 15, 28, 32,37, 47, and 48 are the cases where never re-expanding expands more nodes. From this experiment, we have seen that there is diversity in results. It might happen because we have used inconsistent heuristics.

Figure 6, shows the results of using inconsistent heuristics and various weights. The outcome shows that, for inconsistent heuristics and for various weight (w) values, the never re-expanding policy comes out on top in most cases. Accordingly, if we employ an inconsistent heuristic, we can anticipate that the never re-expanding policy will provide us with a better outcome with fewer node expansions.

Experimenting with always re-expanding and never re-expanding policies on brc202d map: varying the weights (wA*)

Table 2, shows the results of experimenting with always re-expanding and never re-expanding policies on brc202d map. We have observed that regardless of the policy we employ, wA* expands the same number of nodes in the majority of cases. However, the crucial point is that we have observed instances when never re-expanding results in fewer nodes than with re-expanding. We did not see the reverse circumstance. We can say that for this kind of domain we shouldn't apply the re-expanding policy because this particular map has less free spaces and more arbitrary obstacles.

w	without reexpand win	With reexpand win
3	2.597402597	0
5	5.194805195	0
10	11.68831169	0
15	18.18181818	0

Table 2: Experimental Result on map brc202d (wA*)

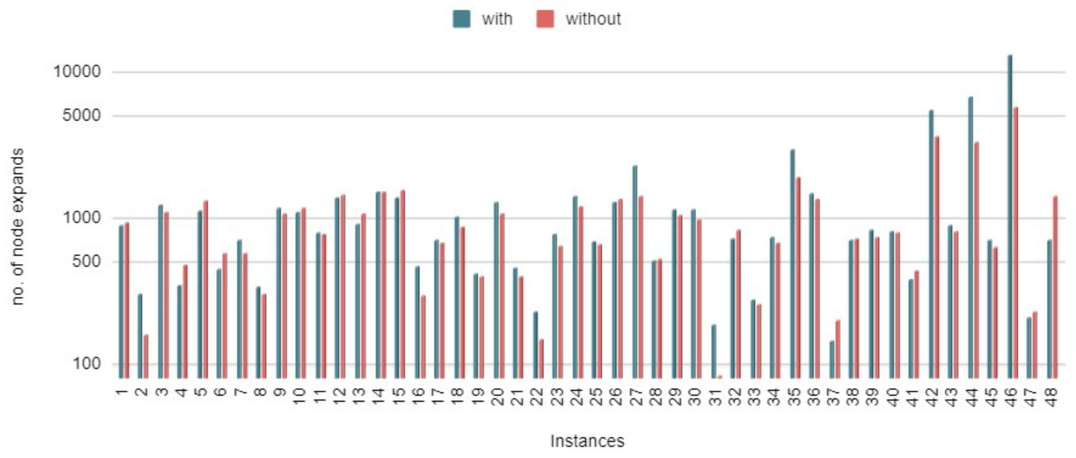

Figure 5: Number of nodes expand with and without re-expanding using inconsistent heuristic
.

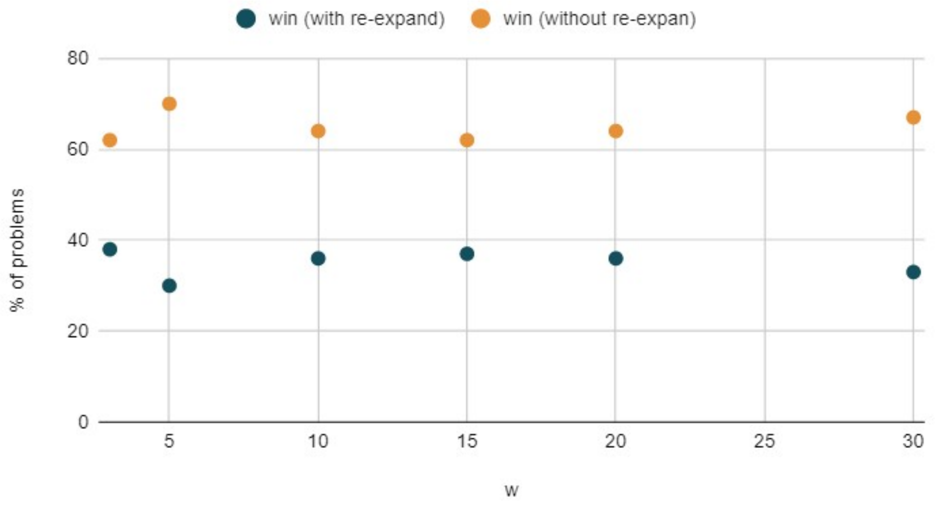

Figure 6: Number of nodes expand with and without re-expanding using inconsistent heuristic
.

Experimenting with always re-expanding and never re-expanding policies (Anytime wA*)

From figure 7 and figure 8, we can observe that, in terms of the quantity of node expansions and execution time, the never re-expanding policy dominates. However, it is not the case in reality. Because most of the time we won't obtain the best results without re-expanding in wA*. Figure 9 displays ten situations in which Anytime wA* failed to find the optimal solution without re-expanding (there are more instances like these 10, but for clear picture we show only 10)

Experimenting with combining always re-expanding and never re-expanding policies (Anytime wA*)

As seen by the results of the preceding experiment, anytime wA* without re-expansions would not converge to the ideal solution. To address this problem, we propose the concept of repairing not re-expanding policy using the idea of anytime repairing A* (Likhachev, Gordon, and Thrun 2003). We experimented with Anytime wA* with no re-expansions will and then moving the not re-expanded nodes back to the OPEN list whenever a new solution is found. Using this combination method takes less time to find the optimal solution. We can see this result in figure 10

Conclusion and Future Work

In this paper, we show that both the always re-expanding and the never re-expanding policies can be advantageous in certain circumstances. Re-expanding the node is supposed to degrade performance and extend additional nodes, according to the majority of earlier research. However, our investigation revealed that this phenomenon only occurs in domains with little open space and a high density of obstacles. If we employ an inconsistent heuristic, we must also

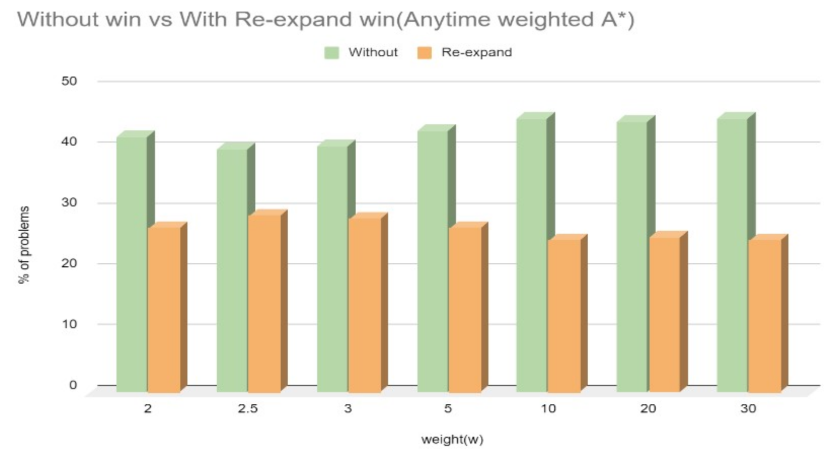

Figure 7: With or Without Re-expanding (anytime wA*)

.

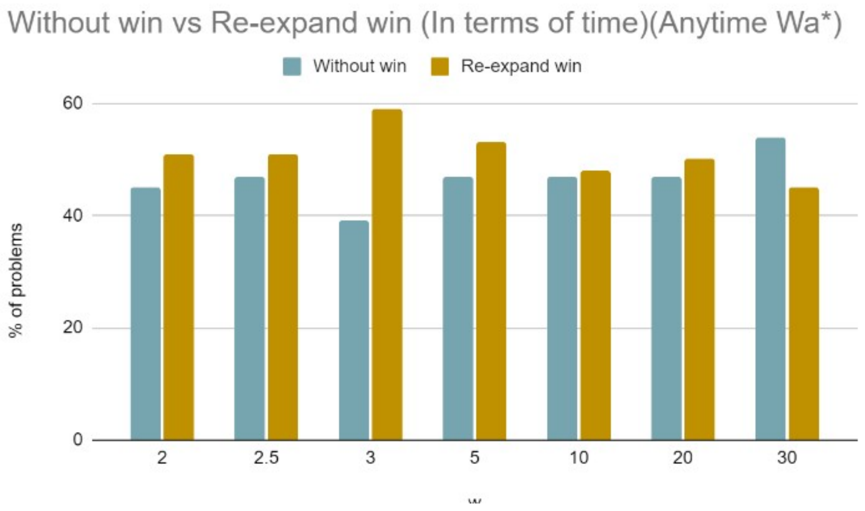

Figure 8: With or Without Re-expanding (in terms of time) (Anytime wA*)

.

employ the never re-expanding policy. But both strategies work equally well where there is more free space and fewer arbitrary obstructions. Therefore, a policy's effectiveness is primarily determined by the domain. Therefore, by using a neural network or machine learning to recognize the domain, we can determine whether we should use re-expand or not. This could be a future extension of this work.

Reference

Pohl, I. 1970. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1(4447): 193–204.

Ariel Felner, R. H. J. S. N. S. Z. Z., Uzi Zahavi. 2011. Inconsistent heuristics in theory and practice. *Artificial Intelligence*, 175: 1570–1603

Uzi Zahavi, J. S., Ariel Felner; and Sturtevant, N. 207. Inconsistent Heuristics. *National Conference on Artificial Intelligence*, 2

Valenzano, R.; Sturtevant, N. R.; and Schaeffer, J. 2014. Worst-Case Solution Quality Analysis When Not Re-Expanding Nodes in Best-First Search. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, 885–892. AAAI Press

Thayer, J.; and Ruml, W. 2010. Anytime Heuristic Search: Frameworks and Algorithms Likhachev, M.; Gordon, G.; and Thrun, S. 2003. ARA*: Anytime A* with Provable Bounds on Sub-Optimality. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, 767–774. Cambridge, MA, USA: MIT Press

https://movingai.com/benchmarks/dao/

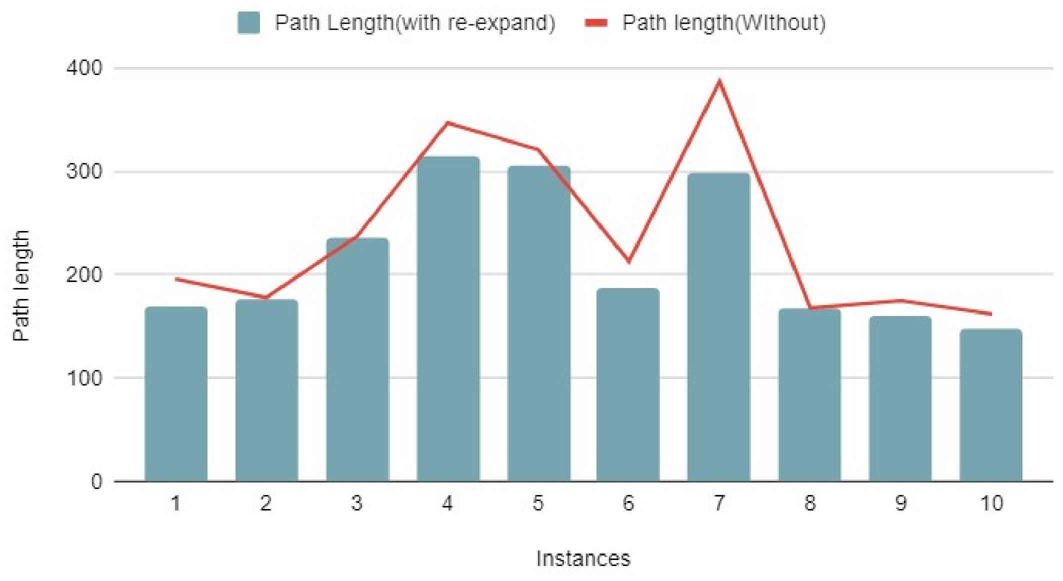

Figure 9: With or Without Re-expanding (in terms of path length) (Anytime wA*)

.

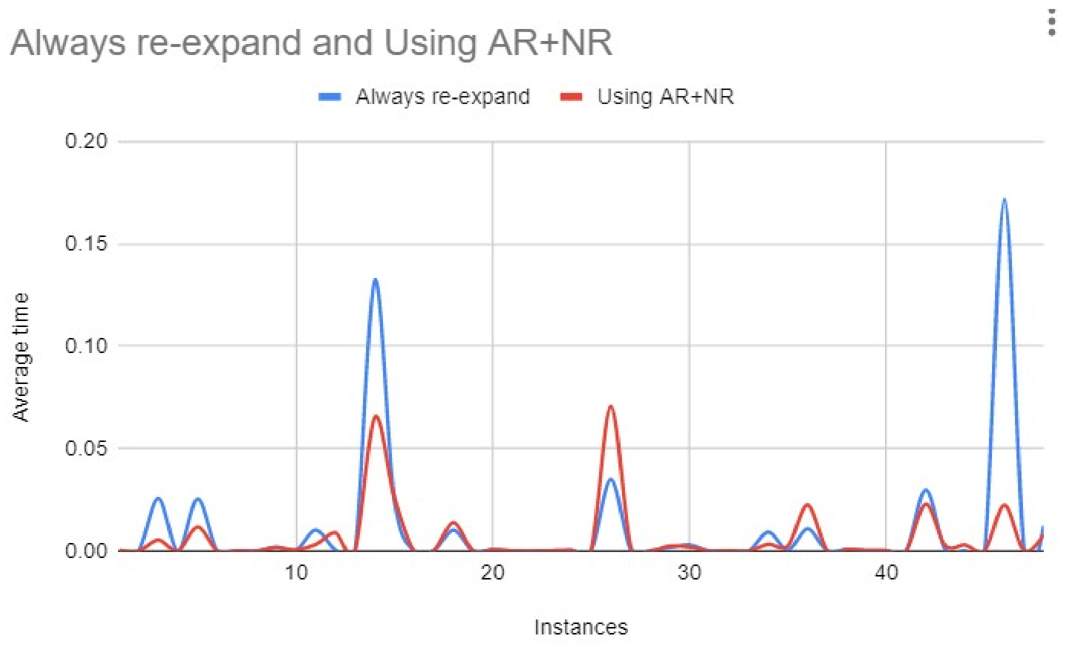

Figure 10: Experiment result using always re-expand and combination of never re-expand and re-expand (in terms of time) (Anytime wA*)

.