



# IJRASET

International Journal For Research in  
Applied Science and Engineering Technology



---

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

---

**Volume: 2      Issue: III      Month of publication: March 2014**

**DOI:**

[www.ijraset.com](http://www.ijraset.com)

Call:  08813907089

E-mail ID: [ijraset@gmail.com](mailto:ijraset@gmail.com)

# Cognizant Accession Subservient to Deadline by Dint of Deadline Aware Search

Shivani Chaudhary  
Department of Computer Science  
DCRUST, Murthal, Haryana (INDIA)

**Abstract:** To perceive provably optimal solutions in many applications of heuristic search insufficient time is available. We contemplate the contract search problem: finding the best solution possible within a given time limit using an interruptible anytime algorithm. Such algorithms return a sequence of improving solutions until interrupted and do not consider the approaching deadline during the course of the search. We propose a new approach, Deadline Aware Search that explicitly takes the deadline into account and attempts to use all available time to find a single high-quality solution.

**Keywords:** Contract search, deadline aware search, heuristic search, best first search.

## I. INTRODUCTION

Heuristic search is an oft employed technique for automated problem solving. Given an admissible and consistent heuristic, A\* search (Hart, Nilsson, and Raphael (1968))[4] finds an optimal solution using the smallest possible number of expansions, up to tie-breaking, of any similarly informed algorithm (Dechter and Pearl (1988))[3]. Unfortunately for many problems of practical interest finding an optimal solution still requires an impractical amount of time. In this paper, we address one attractive approach to this dilemma, *contract search*, in which the objective is to find the cheapest solution possible within a given deadline. There are two real time contract search algorithms & neither performs particularly well in the following evaluation. This may be why the prevailing approach to solving such search problems is to use an interruptible anytime algorithm. While anytime algorithms are applicable to the problem of contract search, they are designed for use in problems where the deadline is unknown. The deadline has no impact on the search order of these algorithms, save for what node will be the last to be expanded. We propose that knowledge of the time remaining

in the search can be used to alter the search order productively

by allocating all search effort towards optimizing a single solution, rather than discarding all but the last one found. In this paper we propose a new algorithm called Deadline Aware Search

(DAS) that is based directly on the objective of contract search: finding the best single solution possible within the deadline. At each iteration the search expands the state that appears to lead to the best solution deemed reachable within the time remaining. Our empirical analysis shows that DAS can compete with and often surpasses previous contract approaches and the leading anytime algorithms on variants of gridworld navigation, the sliding tiles puzzle, and dynamic robot navigation without relying on off-line learning or parameter optimization as previous proposals do.

## II. PREVIOUS WORK

We will first review the anytime approach to search under a deadline. We then review the two previous proposals for contract algorithms before presenting Deadline Aware Search, a new approach to the problem of contract search.

*Anytime Algorithms:* Interruptible anytime algorithms are a class of algorithms that are designed to quickly return a highly suboptimal solution followed by a sequence of solutions of improving quality, eventually converging on optimal. These algorithms are often applied to the problem of search under a deadline because they can be configured to find the first solution very quickly, guaranteeing that some solution will be present at the deadline, and as the deadline is extended the cost of the solutions returned decreases, eventually to optimal. Anytime

## INTERNATIONAL JOURNAL FOR RESEARCH IN APPLIED SCIENCE AND ENGINEERING TECHNOLOGY (IJRASET)

Repairing A\* (ARA\*) performs weighted A\* (Pohl 1973)[8] search to find a starting incumbent solution and then continues searching to find a sequence of improved solutions, eventually converging to the optimal. After each new solution is found the weight used in the search is reduced by some predefined amount, the open list is resorted, & search continues. Problem with the current anytime approaches is that the best performing algorithms are based on bounded suboptimal search, which requires that the bound be set prior to execution. While in some domains there is a single initial bound that performs well over the range of deadlines, there are others in which one setting will perform better for shorter deadlines and another for longer. There is currently no clear way to select a bound based on anything other than training on similar problems and deadlines, or intuition.

- 2) *Time Constrained Search*: Hiraishi, Ohwada, and Mizoguchi (1998)[5] proposed Time Constrained Search, a contract algorithm based on weighted A\*. It attempts to measure search behavior in order to adjust the weight used in weighted A\* in order to meet the deadline while optimizing solution quality. They perform a standard weighted A\* search on  $f_0(s) = g(s) + w \cdot h(s)$ , where  $g(s)$  represents the cost of the path explored thus far,  $h(s)$  is the heuristic estimate of cost-to-go, and  $w$  is a weight factor that they adjust dynamically. They take advantage of the fact that increasing the weight  $w$  generally has the effect of biasing search effort towards states that are closer to goals, reducing solving time. Search behavior is adjusted using search velocity. While their empirical analysis illustrates the quality of solutions found over a range of real-time deadlines (with the contract specified in seconds of computation time), no comparisons were made to previously proposed algorithms. Despite our best efforts to implement and optimize this algorithm we were unable to create a real-time version that was comparable to existing approaches.

- 3) *Contract Search*: Contract Search (Aine, Chakrabarti, and Kumar 2010)[1] attempts to meet the specified deadline by limiting the number of state expansions that can be performed at each depth in the search tree. The algorithm is based around the following insight into search on trees: for an algorithm to expand the optimal goal, it need only expand a single state along an optimal path at each depth. The idea behind Contract Search is to expand only as many states as needed at each depth in order to encounter the optimal solution. We can obviously not know this information a priori. We can, however, assume that the more

states we expand at a given depth, the more likely we are to have expanded the state on the optimal path. Contract search attempts to maximize the likelihood that an optimal solution is found within the deadline by maximizing the likelihood that this optimal state is expanded at each depth, subject to an expansion budget.

### Deadline Aware Search(*starting state, deadline*)

1. *open* {*starting state*}
2. *pruned* {}
3. *incumbent* NULL
4. while (*time*) < (*deadline*)
5. if *open* is non-empty
6.  $d_{max}$  calculate  $d$  bound()
7. *s* remove state from *open* with minimal  $f(s)$
8. if *s* is a goal and is better than *incumbent*
9. *incumbent* *s*
10. else if  $b d(s) < d_{max}$
11. for each child *s*<sub>0</sub> of state *s*
12. add *s*<sub>0</sub> to *open*
13. else
14. add *s* to *pruned*
15. else (\* *open* is empty \*)
16. recover *pruned* states(*open, pruned*)
17. return *incumbent*

### Recover Pruned States(*open, pruned*)

18. *exp* estimated expansions remaining
19. while *exp* > 0 and *pruned* is non-empty loop
20. *s* remove state from *pruned* with minimal  $f(s)$
21. add *s* to *open*
23. *exp* = *exp* -  $b d(s)$

Figure 1: Pseudo-code sketch of Deadline Aware Search

The largest contract considered by (Aine, Chakrabarti, and Kumar 2010)[1] is 50,000 expansions. In our evaluation, we will be considering search deadlines of up to a minute, which for our benchmark domains could mean more than five million states. This is problematic because the time and space complexity of computing these values grows quadratically in the size of the contract. Aine (2011)[2] suggested approximating the table by only considering states in chunks, rather than a single state at a

# INTERNATIONAL JOURNAL FOR RESEARCH IN APPLIED SCIENCE AND ENGINEERING TECHNOLOGY (IJRASET)

time. This cuts down on the size of the table and the number of computations we need to perform to compute it. In the results presented below, the resolution was selected so that the tables needed could be computed within 8 GB of memory. Computing the table typically took less than eight hours per domain, although a new table must be computed for each considered deadline.

### III. PROPOSED WORK

#### *Deadline Aware Search*

We now present a new approach for the contract search problem, the called Deadline Aware Search (DAS). Unlike any time search algorithms that do not alter their search strategy in reaction to the approaching deadline, DAS reacts to the approaching deadline during search. We begin by presenting a general overview of the algorithm and its behavior. We then discuss the estimation of two quantities needed by the algorithm: the maximum achievable search depth  $d_{max}$  and distance to the cheapest solution beneath a node  $\hat{d}_{cheapest}(s)$ . Finally, we discuss DAS's technique for recovering from situations in which it estimates that no goal is reachable given the current search behavior. DAS is a simple approach, derived directly from the objective of contract search. It expands, among all the states leading to solutions deemed reachable within the time remaining, the one with the minimum  $f(s)$  value. Pseudocode of the algorithm is presented in Figure 1. The open list is first initialized with the starting state and then the search proceeds to expand nodes from the open list until either the search time expires or the open list is empty (indicating that there is no solution deemed reachable). At each iteration of the algorithm, the state with minimal  $f(s)$  is selected for expansion and the current maximum reachable distance,  $d_{max}$ , is calculated. If the distance to this state's best goal  $\hat{d}_{cheapest}(s)$  is less than  $d_{max}$ , it is expanded and its children are added to the open list. Otherwise, it is added to the pruned list and the search will select the next best node for expansion.

#### *A. Calculating $d_{max}$*

One could argue that all remaining search effort should be put towards finding the best possible solution under the single state in the search space estimated to be best and therefore the  $d_{max}$  value used to prune states should be equal to the estimated number of expansions possible in the time remaining. In practice, however, this approach renders the value of  $d_{max}$  meaningless for

most of the search in which the number of expansions will often be far larger than the estimated length of the path to any particular solution. If the number of expansions allowed were close to the length of the path to a solution then one could hardly consider performing any type of search other than depth-first! Pilot empirical studies have confirmed that such an interpretation of  $d_{max}$  produces unreasonable behavior. To understand the true number of expansions it will take to find a solution under a particular state, one must consider the behavior of the search itself. In a best-first search, it is not typical that a single path will be followed from the starting state to the optimal solution. Often, due to error in the heuristic, when a state is expanded the  $f$  value of its best child state  $s_{bc}$  is greater than  $f(s)$ . When this occurs it is possible that there exists some other state in the open list  $s_0$  such that  $f(s_0) < f(s_{bc})$ . If the search continues unhindered, it will stop exploring the path leading to  $s_{bc}$  and start examining the path currently leading to  $s_0$ . This behavior is a phenomenon that we call *search vacillation*, where multiple partial solutions are being explored during the search. One way to measure this vacillation, is a concept we call *expansion delay*. During the search, the number of expansions performed is tracked,  $e_{curr}$ , expansions.

#### *B. Pruning On $\hat{d}(s)$*

DAS makes use of a heuristic  $\hat{d}_{cheapest}(s)$  that estimates the length of the path to the cheapest goal state under a particular state  $s$ . For unit-cost domains this heuristic is the same as the standard cost-to-go  $h(s)$ . For non-unit cost domains it can often be constructed similarly to  $h(s)$  by estimating the same cheapest path while replacing all actions costs with 1. We do not require that  $\hat{d}_{cheapest}(s)$  be admissible or even consistent, in fact, it is preferable for  $\hat{d}_{cheapest}(s)$  to act as a differentiator between different paths by more accurately accounting for heuristic error. We employ the path-based correction model of Thayer, Dionne, and Ruml (2011) to calculate a corrected heuristic  $\hat{d}_{cheapest}(s)$ . Briefly, this correction model uses error experienced at each expansion:  $\bar{e}_{d}(s) = \hat{d}_{cheapest}(s) - \hat{d}_{cheapest}(p(s)) + 1$ , where  $p(s)$  represents the parent of state  $s$ . The mean single step error encountered so far  $\bar{e}_{d}(s)$  is tracked for each partial solution and is used as an estimator of the average single step error remaining from the end of that partial solution to the corresponding goal state.

#### *C. Search Recovery*

# INTERNATIONAL JOURNAL FOR RESEARCH IN APPLIED SCIENCE AND ENGINEERING TECHNOLOGY (IJRASET)

It can be the case that the effects of pruning are not enough to keep the  $\hat{d}(s)$  of at least one state in the open list below the value of  $d_{max}$ . In these cases, DAS will prune all states from the open list as “unreachable”. This is an indication that despite the best efforts of the algorithm, the amount of vacillation remaining in the search due to competing states on the open list will result in no further solutions being reached. To recover from this, we first estimate the number of expansions possible in the time remaining  $exp$ . States are then selected from the pruned list in order of minimal  $f(s)$  values such that the sum of  $\hat{d}(s)$  for all states inserted is approximately equal to  $exp$ . In order to guarantee that at least one state is placed back into open at each recovery we insert states into open until the sum of  $\hat{d}(s)$  first exceeds  $exp$ . The intention is that only the best set of states that would be reachable regardless of the search behavior are kept. Because the open list has changed so drastically, the previous estimation of average expansion delay  $\bar{e}$  is no longer relevant and the running average is reset. This allows the search to continue and measure the new local behavior after the recovery. The same settling time used at the start of the search must be applied, during which  $d_{max}$  is not calculated or used.

## IV. CONCLUSION

We have proposed a new method of measuring search behavior, expansion delay, that can be used as an indicator of the level of vacillation present in the search due to heuristic error leading to competition between different paths on the open list. Using this measure we have constructed a very simple and general approach to the problem of heuristic search under deadlines: Deadline Aware Search. DAS appears to be the first effective contract heuristic search algorithm, showing improvements over ARA\* and RWA\* in several domains using real time as deadlines. Our approach also has the benefit of being parameterless, learning necessary information on-line, while previous approaches required either parameter optimization or off-line training and pre-computation. The problem of heuristic search under real-time deadlines is of great importance in practice and yet few algorithms have been proposed for that setting. While anytime methods are certainly applicable, they are really designed to address the problem of search when the deadline is unknown. While simple, our approach illustrates that knowledge of the termination deadline can improve performance for contract search.

## ACKNOWLEDGEMENT

I gratefully acknowledge support from the CSI professors for their responses against the doubts that congested the mind as a whole & especially to Rakesh Chawla for his responses to our inquiries about contract search & beam search.

## REFERENCES

- [1] Aine, S.; Chakrabarti, P.; and Kumar, R. 2010. Heuristic search under contract. *Computational Intelligence* 26(4):386–419.
- [2] Aine, S. 2011. Personal communication.
- [3] Dechter, R., and Pearl, J. 1988. The optimality of A\*. In Kanal, L., and Kumar, V., eds., *Search in Artificial Intelligence*. Springer-Verlag. 166–199.
- [4] 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.
- [5] Hiraishi, H.; Ohwada, H.; and Mizoguchi, F. 1998. Time constrained heuristic search for practical route finding. In *Pacific Rim International Conferences on Artificial Intelligence*.
- [6] Korf, R. E. 1990. Real-time heuristic search. *Artificial Intelligence* 42:189–211.
- [7] Likhachev, M.; Gordon, G.; and Thrun, S. 2003. ARA\*: Anytime A\* with provable bounds on sub-optimality. In *Proceedings of the Seventeenth Annual Conference on Neural Information Processing Systems (NIPS-03)*.
- [8] 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.
- [9] Richter, S.; Thayer, J. T.; and Ruml, W. 2009. The joy of forgetting: Faster anytime search via restarting. In *Symposium on Combinatorial Search*.
- [10] Ruml, W., and Do, M. B. 2007. Best-first utility-guided search. In *Proceedings of IJCAI-07*, 2378–2384.
- [11] Thayer, J. T., and Ruml, W. 2010. Anytime heuristic search: Frameworks and algorithms. In *SoCS 2010*.
- [12] Thayer, J. T.; Dionne, A.; and Ruml, W. 2011. Learning inadmissible heuristics during search. In *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS-11)*.



10.22214/IJRASET



45.98



IMPACT FACTOR:  
7.129



IMPACT FACTOR:  
7.429



# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24\*7 Support on Whatsapp)