G-value Plateaus: A Challenge for Planning

J. Benton[†] and **Kartik Talamadupula**[†]

Patrick Eyerich[‡] and **Robert Mattmüller**[‡] and **Subbarao Kambhampati**[†]

[†] Dept. of Computer Science and Eng.
Arizona State University
Tempe, AZ 85287 USA
{j.benton,krt,rao}@asu.edu

[‡] Department of Computer Science
University of Freiburg
Freiburg, Germany
{eyerich,mattmuel}@informatik.uni-freiburg.de

Abstract

While the string of successes found in using heuristic, best-first search methods have provided positive reinforcement for continuing work along these lines, fundamental problems arise when handling objectives whose value does not change with search operations. An extreme case of this occurs when handling the objective of generating a temporal plan with short makespan. Typically used heuristic search methods assume strictly positive edge costs for their guarantees on completeness and optimality, while the usual "fattening" and "advance time" steps of heuristic search for temporal planning have the potential of resulting in "g-value plateaus". In this paper we point out some underlying difficulties with using modern heuristic search methods when operating over g-value plateaus and discuss how the presence of these problems contributes to the poor performance of heuristic search planners. To further illustrate this, we show empirical results on recent benchmarks using a planner made with makespan optimization in mind.

Introduction

Search space topology has received significant attention in planning; efforts such as those made by Hoffmann (2005) have, for example, identified the problem of h-value plateaus, regions of the search space where h-values do not change. A related, but less recognized problem within the planning community are g-value plateaus, in which search operations do not increase the g-value over large regions. While h-value plateaus occur because of the imperfect or myopic nature of heuristics, g-value plateaus can trace their roots to a more basic mismatch between the search operations and objectives. Specifically, the children (and descendants) of a search node may not necessarily have a higher g-value than the node. There are many objectives in planning for which standard formulations tend to lead to g-value plateaus. Perhaps the poster child of such objectives is *makespan* minimization using normal state-space (or decision-epoch (Cushing et al. 2007)) search formulations. It is easy enough to see that the operation of adding an action to a plan does not necessarily increase its makespan.[1]

[1]Another example is partial order planning search, where many operations, such as establishment, do not increase cost.

At first glance, g-value plateaus seem to run afoul of the "positive edge cost" assumption that is the textbook sufficiency condition for the completeness/termination of A^* search. Fortunately, while the absence of g-value plateaus is sufficient to guarantee completeness, their presence does not necessarily lead to incompleteness as long as there are no infinitely long bounded-value (zero-cost) paths. This is often of little consolation however as g-value plateaus can, and do, significantly inhibit A^*-search especially when coupled with non-pathological heuristics (i.e., where $h(s) < h^*(s)$). This issue is by no means unique to planning: the work on treewidth computation using best-first search (Dow and Korf 2007) also highlights problems similar to those encountered in makespan minimization (although they do not connect it to the broader g-value plateau problem).

In this paper, we aim to call the community's attention to the problem of g-value plateaus and the need for techniques to handle them. In order to do this effectively, we need to answer two natural questions: (i) why has the planning literature not paid much attention to the deleterious effects of g-value plateaus? and (ii) just how much of an effect do g-value plateaus have on the efficiency of the underlying search? In the following, we will answer both these questions, specifically in the context of finding plans of low *makespan*. For the first, we shall show how existing systems may overlook the presence of g-value plateaus either (a) because they have usually focused on multi-objective search (e.g., Sapa (Do and Kambhampati 2003)) where the search operations are not mismatched with respect to at least one of the objectives or (b) because search space representations must attribute a portion of the minimum cost through a state to the h-value (as against the g-value) to ensure certain properties hold. Turning to question (ii), we will quantify the effect of g-value plateaus on search both in theoretical and empirical terms. In the case of optimal planning we show that g-value plateaus will necessarily cause expansion of all states within the plateau that lead to optimal solutions. For suboptimal planning, we will state criteria for when g-value plateaus can force state expansions and empirically confirm that g-value plateaus can cause poor performance. We conclude the paper with some suggestions and show how "pseudo-multi-objective" search that looks at "cost" objectives even when the main focus is makespan can improve performance without significantly degrading plan quality.

Case Study: Minimizing Makespan

It can be shown that typical temporal state-space planners working to optimize makespan can suffer from plateaus over the objective function. However, this fact may be overlooked at first glance due to their state space models and how they attribute h- and g-values. We present an evaluation of current approaches that shows how plateaus over the objective function can appear.

Temporal Planning Background

Many existing forward and regression heuristic search temporal planners use a decision epoch time point as their g-value.[2] In usual forward chaining search methods (Bacchus and Ady 2001; Do and Kambhampati 2003; Eyerich, Mattmüller, and Röger 2009), this is the "current time" for actions to begin execution; the regression planner TP4 (Haslum and Geffner 2001), on the other hand, uses a different style of search where the g-value represents the time from the goal to a state's decision epoch. This entails keeping time points of where "currently executing" actions end (progression) or begin (regression). In the following, we call these *action time points* to simplify discussion.

State Evaluation

A usual problem for makespan optimization in state space temporal planning relates to adding long actions that must run in parallel with a series of shorter actions to obtain a high quality solution. Often in these cases, many permutations of the shorter actions can be considered. That is, there is long action giving a constant minimum bound on the makespan for a state, a value different than the current decision epoch time point. In the usual A^* evaluation function of $f(s) = g(s) + h(s)$ for a state s, this constant bound can be on either $g(s)$ or $h(s)$. In terms of the discussed planning methods, we call the decision epoch $g_t(s)$ (with a corresponding heuristic $h_t(s)$) and distinguish it from the known minimum makespan, which we call $g_m(s)$. More formally, given a state s and a maximum distance action time point from $g_t(s)$, $t_{max}(s)$ we define $g_m(s) = \max(g_t(s), t_{max}(s))$ (see Figure 1). Note that we cannot (and should not) easily use this view of state evaluation in practice, as it can produce complications in representing the state space (for instance, $h_m(s) = 0$ is not enough for a goal test). Ultimately, since $h_t(s)$ defines a minimum value that can be found with constant computational time, we can perform this transfer of values from h to g. For this reason the transfer serves to simplify discussion and analysis to considering only g-value plateaus whenever such minimum bounds exist in the calculation of h.

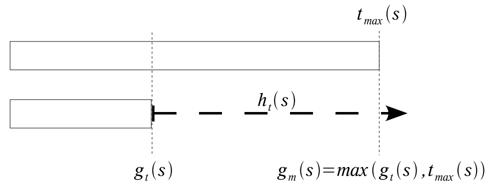

Figure 1: A state s in a (progression) temporal heuristic search planner.

Plateaus on g

In temporal planning problem benchmarks, adjacent search states with equal makespan (as defined by g_m) can occur in abundance since it is often possible to execute extraneous actions in parallel with useful ones without changing the final plan makespan, leading to g-value plateaus. Informally, g-value plateaus are a tree structure within the search space where the evaluation values of states do not change from ancestor to descendant. They are inherently *local* in nature. We discuss plateaus on g in both the optimal and suboptimal contexts but first give them a more formal definition.

Definition 1. *A g-value plateau with respect to state s is the partial subtree $T[s]$ where s is the root and $T[s]$ includes all immediate descendants s' where $g(s) = g(s')$ and all states in the g-value plateau $T[s']$.*

In the worst case, when all states in the g-value plateau $T[s]$ have a constant heuristic value, upon expanding a state s, A^* will eventually expand all states $s' \in T[s]$. This implies that better heuristics or tie-breaking rules may significantly improve matters. It turns out that this is not always enough. In optimal planning, multiple paths to equally valued states can cause expansion of large portions of the g-value plateau.

Theorem 1. *Given a g-value plateau $T[s]$ (of a state s) that does not contain a goal state and an admissible but non-pathological heuristic function h, upon expanding s during the search, A^* will eventually expand all state $s' \in T'[s]$ before reaching a goal state, where $T'[s] \subseteq T[s] \setminus \{s\}$ is the set of states that can be expanded to an optimal path.*

Proof. Let g^* be the optimal solution value. For any state $s' \in T'[s]$: $h^*(s) = h^*(s')$, since $g(s) + h^*(s) = g(s') + h^*(s') = g^*$ and $g(s) = g(s')$. As h is an admissible and non-pathological heuristic function, $h(s') < h^*(s')$ which implies $f(s') = g(s') + h(s') = g(s) + h(s') < g(s) + h^*(s') = g(s) + h^*(s) = g^*$. Therefore, A* must expand s' before finding an optimal solution. □

A similar problem also occurs in the case of finding suboptimal plans using any heuristic (e.g., inadmissible or weighted heuristics) in the A^* framework. Specifically, we can characterize a portion of the g-value plateau that must be explored.

Proposition 1. *Consider a path $P = (s_0, \ldots, s_g)$ from an initial state s_0 to a solution state s_g found with an A^* search using an (inadmissible) heuristic h. Let s_i be a state in the path such that its g-value plateau, $T[s_i]$, does not contain s_g, s_j $(j > i)$ be the first state s.t. $g(s_j) > g(s_i)$ (and therefore is outside the plateau), and $s_{max} = \arg\max_{s \in (s_j, \ldots, s_g)}\{f(s)\}$. Then, upon expanding*

[2] We are focusing on planners that handle temporally "simple" models, i.e., those that cannot generally handle problems with *required concurrency*.

s_i, before reaching the goal state A^* will have eventually expanded all states s in the plateau that can be reached from s_i through at least one path with all *states s' s.t. $f(s') < f_{max} = f(s_{max})$.*

Proof. As all the states s' along the path from s_i to s satisfy $f(s') < f_{max}$, they will be pulled off the queue before the state s_{max}. □

This proposition implies that there is a portion of the g-value plateau with states s where $h(s) < h_{max} = f_{max} - g(s_i)$ that eventually must be explored. In other words, the heuristic function must return a value greater than the bound h_{max} in order to prune this search space, underscoring that only the h-value can guide search over g-value plateaus.

Empirical Confirmation

We designed experiments to analyze the search spaces of recent temporal planning benchmark problems for g-value plateaus on makespan. To enable this analysis, we designed a new (inadmissible) makespan heuristic (h_m) on top of the context-enhanced additive heuristic used in the Temporal Fast Downward (TFD) planner (Eyerich, Mattmüller, and Röger 2009).

The heuristic uses the same method as TFD for determining sets of actions required to reach the goal. In the makespan heuristic we make use of the causal constraints between actions (assuming TGP semantics (Smith and Weld 1999)) as detected during the extraction of the heuristic plan. These constraints, together with duration constraints between when action begin and end points, are encoded in a Simple Temporal Network (STN). The makespan of the schedule produced by solving the STN is then returned as the heuristic estimate.

Table 1 summarizes the results of the empirical analysis on the 6th International Planning Competition (IPC-2008) domains. Our experiments were run on an Intel Xeon processor running at 2.66 Ghz at a 10 minute time limit running using SuSE Linux. Unlike the original TFD, the planner neither performs an anytime search nor uses "preferred operators". Note that the values r_g and r_f in the table include runs where no solution was found (but a sample of the search space could still be taken).

We define the function $g_c(s)$ as the sum of all action durations chosen at s, $h_c(s)$ as the heuristic on the sums of durations on the "actions-to-go", $h_m(s)$ as our new heuristic, and $g_m(s)$ as described previously (i.e., the makespan of s up to the longest running action). The results show remarkably large portions of the search space with g-value plateaus on makespan in all domains where $f_m = g_m + h_m$ is used. Notice that, except in openstacks-adl and openstacks-strips, f-value plateaus increase significantly with g-value plateaus (an expected result). Contrasting with $f_c = g_c + h_c$, it is apparent that using summed durations improves coverage of problems solved significantly, though the quality of the solutions is reduced. Note that it is possible to get equal g-values on parents and children using g_c in our search with the "advance time" operation, which does not add any actions.

Steps Toward a Solution

By now we have accomplished the main aim of this paper, which is to bring the challenge to the foreground. In this section, we discuss steps towards handling this challenge and share results on one promising idea.

The problem with g-value plateaus is that they can induce search that is worse than standard breadth-first search, a phenomenon that can occur in A^* search with any non-uniform cost values on transitions. g-value plateaus may be seen as a special case of this and offer a step in identifying such "problem" regions of the search space. For instance, one possibility of handling these situations is to find "exit points" to any state beyond the g-value plateau. While it is generally impossible to know f_{max}, we can at least identify g-value plateaus, areas we know have strong dependency on the behavior of the heuristic. In particular, we consider finding a set of exit points that are diverse in their potential reachability across a plateau (c.f., Srivastava et al. (2007)). Of course, such techniques would likely need to be *anytime* in nature for complete and optimal planning.

Another possibility is to try to remove the plateaus by using equivalence classes between states. Such analysis would likely be domain-dependent but if done properly may collapse g-value plateaus. Related to this, in planning for makespan, we can consider using causal analysis to avoid adding extraneous actions. However, this is impractical in general (though approximation methods may be used).

One may consider the idea of adding a small increase in g-values between a parent and child when there would otherwise be zero cost. Indeed, this approach has been applied in the case of planning with action costs (Richter and Westphal 2008; Keyder and Geffner 2008; Benton, Do, and Kambhampati 2009). However, this is unlikely to succeed in general cases. Small increases, given a large enough f_{max}, would exhibit the same behavior. Instead, we study techniques inspired by Sapa and Temporal Fast Downward (TFD) in the temporal planning setting. These planners found success when performing "pseudo-multi-objective" searches.

We developed an approach that applies an additional heuristic cost related both to the makespan (the objective function with g-value plateaus) and the number of search operations left. Specifically, we apply a weighted "cost" heuristic value, $h_c(s)$, which sums the durations of $h_m(s)$ (i.e., the "makespan-to-go") and use the evaluation function $g_m(s) + h_m(s) + w * h_c(s)$. As we will see next, this technique finds some success. The solution differs from those of Sapa and TFD in that we include a state evaluation over makespan while adding a cost evaluation on duration, whereas Sapa uses makespan to calculate its fundamentally cost-based heuristic and TFD only uses the sum of durations as an estimate of makespan. From this perspective the technique is more related to the revised dynamically weighted A^* approach by Thayer and Ruml (2009). While such techniques improve performance, they may not work well in every domain with g-value plateaus, such as problems with known solution depths (for example, best-first search for treewidth (Dow and Korf 2007)).

	$\mathbf{f_c = g_c + h_c}$				$\mathbf{f_m = g_m + h_m}$				$\mathbf{f_{mw} = g_m + h_w}$			
Domain	r_g	r_f	cov	qual	r_g	r_f	cov	qual	r_g	r_f	cov	qual
crewplanning-strips	0.03	0.55	11	6.82	0.98	0.83	4	4.00	0.95	0.09	12	11.99
elevators-numeric	0.06	0.03	4	2.41	0.57	0.27	2	2.00	0.48	0.05	4	3.78
elevators-strips	0.07	0.05	3	1.70	0.53	0.25	3	2.98	0.44	0.04	4	3.92
openstacks-adl	0.15	0.89	30	17.80	1.00	0.88	8	7.58	0.92	0.02	30	28.00
openstacks-strips	0.14	0.88	30	17.12	0.67	0.29	27	20.93	0.71	0.06	30	25.39
parcprinter-strips	0.16	0.08	12	5.68	0.90	0.37	6	5.31	0.76	0.09	6	5.00
pegsol-strips	0.17	0.09	25	18.77	0.85	0.25	22	21.15	0.82	0.08	26	24.04
sokoban-strips	0.28	0.16	11	10.18	0.78	0.32	10	10.00	0.77	0.10	10	9.83
transport-numeric	0.23	0.06	2	1.26	0.74	0.48	3	3.00	0.58	0.09	3	2.78
woodworking-numeric	0.08	0.12	18	14.30	0.72	0.26	18	16.91	0.55	0.04	19	17.64
overall	0.09	0.21	146	96.04	0.84	0.50	103	93.86	0.68	0.07	144	132.37

Table 1: A comparison of the aggregate fraction of zero-cost search operations (r_g), the fraction of f-valued children with equal value to parent (r_f), problems solved (cov) and plan quality according to the IPC-2008 measure (qual, where higher is better compared against the IPC "reference" plans when available) on the IPC-2008 benchmark domains in the temporal satisficing track on $f_c = g_c + h_c$, $f_m = g_m + h_m$ and $f_{mw} = g_m + h_w$.

Empirical Evaluation

In addition to confirming the existence of and problems with g-value plateaus, we ran experiments to test our simple improvement that combines a makespan and weighted cost heuristic using $h_w = h_m(s) + 0.2 * h_c(s)$. With these functions we ran A^* as before but over the evaluation function $f_{mw}(s) = g_m(s) + h_w(s)$.

Our experiments on f_{mw} show that, despite the g-value plateaus that still remain in the search space, performance improves dramatically from f_m (see Table 1). In fact, in most instances, openstacks-adl in particular, the planner appears to be achieving the best of both worlds, with a high number of problems solved with high quality (and our total quality reflects this increase). Also, by adding a weighted h_c, in most cases the search spaces have a significant decrease in f-value plateaus compared with f_c and f_m.

Conclusion

This paper points out the problem of g-value plateaus and shows how they can lead to poor search performance. We focused particularly on planning for quality solutions using makespan as a prime example of where the problem occurs and studied one solution to this problem. Our empirical results show that, when facing g-value plateaus, we can improve performance significantly using a "pseudo-multi-objective" method that includes a heuristic on a related, but different objective. However, this approach is not optimal and we believe that techniques that focus on identifying g-value plateaus may yield more principled and general approaches compatible with both optimal and suboptimal search.

Acknowledgments: We extend our thanks to Tuan Nguyen, William Cushing and Rong Zhou for their helpful discussions. Many thanks also go to Wheeler Ruml, Jordan Thayer, Sofia Lemons and the anonymous reviewers for their excellent comments. This research is supported in part by ONR grants N00014-09-1-0017, N00014-07-1-1049, the NSF grant IIS-0905672, by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS) and by the EU as part of the Integrated Project CogX (FP7-ICT-2xo15181-CogX).

References

Bacchus, F., and Ady, M. 2001. Planning with resources and concurrency: A forward chaining approach. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 417–424.

Benton, J.; Do, M.; and Kambhampati, S. 2009. Anytime heuristic search for partial satisfaction planning. *Artificial Intelligence Journal* 173(5-6).

Cushing, W.; Kambhampati, S.; Mausam; and Weld, D. S. 2007. When is temporal planning really temporal? In *Proceedings of the 20th IJCAI*, 1852–1859.

Do, M., and Kambhampati, S. 2003. Sapa: A scalable multi-objective metric temporal planner. *Journal of Artificial Intelligence Research (JAIR)* 20:155–194.

Dow, P. A., and Korf, R. E. 2007. Best-first search for treewidth. In *Proceedings of the 22nd Conference on Artificial Intelligence*.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2009. Using the context-enhanced additive heuristic for temporal and numeric planning. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*.

Haslum, P., and Geffner, H. 2001. Heuristic planning with time and resources. In *Proceedings of the 6th European Conference on Planning*.

Hoffmann, J. 2005. Where 'ignoring delete lists' works: Local search topology in planning benchmarks. *JAIR* 24:685–758.

Keyder, E., and Geffner, H. 2008. The FF(h_a) planner for planning with action costs. In *Proceedings of the International Planning Competition (IPC)*.

Richter, S., and Westphal, M. 2008. The LAMA planner. using landmark counting in heuristic search. In *Proceedings of the IPC*.

Smith, D. E., and Weld, D. S. 1999. Temporal planning with mutual exclusion reasoning. In *Proceedings of the 16th IJCAI*, 326–337.

Srivastava, B.; Kambhampati, S.; Nguyen, T.; Do, M.; Gerevini, A.; and Serina, I. 2007. Domain independent approaches for finding diverse plans. In *Proceedings of the 20th IJCAI*.

Thayer, J. T., and Ruml, W. 2009. Using distance estimates in heuristic search. In *Proceedings of the 19th ICAPS*.