Cerberus: Red-Black Heuristic for Planning Tasks with Conditional Effects Meets Novelty Heuristic and Enhanced Mutex Detection

Michael Katz
IBM Research
Yorktown Heights, NY, USA
michael.katz1@ibm.com

Abstract

Red-black planning is the state-of-the-art approach to satisficing classical planning. A planner Mercury, empowered by the red-black planning heuristic, was the runner-up of the latest International Planning Competition (IPC) 2014, despite the trivial handling of conditional effects by compiling them away. Conditional effects are important for classical planning and required in many domains for efficient modeling. Another recent success in satisficing classical planning is the Novelty based heuristic guidance. When novelty of heuristic values is considered, search space is partitioned into novelty layers. Exploring these layers in the order of their novelty considerably improves the performance of the underlying heuristics. Yet another recent success relates to the translation of planning tasks from the input PDDL language to a grounded multi-valued variable based representation, such as SAS+. Recent methods of invariants synthesis allow for deriving richer SAS+ representations.

We herein present a satisficing classical planner which we baptize Cerberus, that incorporates these three recent improvements. It starts by performing enhanced mutex detection to derive a SAS+ planning task with conditional effects. Then, it performs best first search of various greediness, exploiting red-black planning heuristic with a direct handling of conditional effects and using such red-black heuristic as a base for a novelty heuristic.

Introduction

Delete relaxation heuristics have played a key role in the success of satisficing planning systems (Bonet and Geffner 2001; Hoffmann and Nebel 2001; Richter and Westphal 2010). A well-known pitfall of delete relaxation is its inability to account for repetitive achievements of facts. It has thus been an actively researched question from the outset how to take some deletes into account, e.g. (Fox and Long 2001; Gerevini, Saetti, and Serina 2003; Helmert 2004; Helmert and Geffner 2008; Baier and Botea 2009; Cai, Hoffmann, and Helmert 2009; Haslum 2012; Keyder, Hoffmann, and Haslum 2012). Red-black planning framework (Domshlak, Hoffmann, and Katz 2015), where a subset of red state variables takes on the relaxed value-accumulating semantics, while the other black variables retain the regular semantics, introduced a convenient way of interpolating between fully relaxed and regular planning.

Katz, Hoffmann, and Domshlak (2013b) introduced the red-black framework and conducted a theoretical investigation of tractability. Following up on this, they devised practical red-black plan heuristics, non-admissible heuristics generated by repairing fully delete-relaxed plans into red-black plans (Katz, Hoffmann, and Domshlak 2013a). Observing that this technique often suffers from dramatic over-estimation incurred by following arbitrary decisions taken in delete-relaxed plans, Katz and Hoffmann (2013) refined the approach to rely less on such decisions, yielding a more flexible algorithm delivering better search guidance. Subsequently, Katz and Hoffmann (2014b) presented a red-black DAG heuristics for a tractable fragment characterized by DAG black causal graphs and devise some enhancements targeted at making the resulting red-black plans executable in the real task, stopping the search if they succeed in reaching the goal. Red-black DAG heuristics are in the heart of the Mercury planner (Katz and Hoffmann 2014a), the runner-up of the sequential satisficing track in the latest International Planning Competition (IPC 2014). All aforementioned work on red-black planning, however, handles the SAS+ fragment without conditional effects, despite of conditional effects being a main feature in the domains of IPC 2014. The planner Mercury that favorably participated in IPC 2014, handles conditional effects by simply compiling them away (Nebel 2000). Obviously, the number of actions in the resulted planning tasks grows exponentially, and thus such straight forward compiling away does not scale well. Nebel (2000) presents an alternative compilation, that does not lead to an exponential blow-up in the task size. This compilation, however does not preserve the delete relaxation. Thus, several delete relaxation based heuristics were adapted to natively support conditional effects (Haslum 2013; Röger, Pommerening, and Helmert 2014). Recently, Katz (2018) has shown that the fragment of red-black planning characterized by DAG black causal graphs remains tractable in the presence of conditional effects, extending the existing red-black planning heuristics to natively handling conditional effects.

Search-boosting and pruning techniques have considerably advanced the state-of-the-art in planning as heuristic search (Richter and Helmert 2009; Richter and Westphal 2010; Xie et al. 2014; Valenzano et al. 2014; Domshlak, Katz, and Shleyfman 2013; Lipovetzky and Geffner 2012).
One such technique is based on the concept of novelty of a state, where the search procedure prunes nodes that do not qualify as novel. The concept has been successfully exploited in classical planning via $SIW^+$ and $DFS(i)$ search algorithms and in heuristic search, in conjunction with helpful actions (Lipovetzky and Geffner 2012; 2014; 2017), and in blind state-space search for deterministic online planning in Atari-like problems (Lipovetzky, Ramirez, and Geffner 2015), where it was later generalized to account for rewards (Shleyfman, Tuisov, and Domshlak 2016; Jinnaei and Fukunaga 2017). The latter work, although applied to Atari-like problems, is valid for planning with rewards in general, when rewards are defined on states. Consequently, (Katz et al. 2017) brought the concept of novelty back to heuristic search, adapting the novelty definition of Shleyfman, Tuisov, and Domshlak (2016) to a novelty of a state with respect to its heuristic estimate. The new novelty notion was no longer used solely for pruning search nodes, but rather as a heuristic function, for node ordering in a queue. However, since such heuristics are not goal-aware, Katz et al. (2017) use the base goal-aware heuristic as a secondary (tie-breaking) heuristic for node ordering.

In this work we construct a planner Cerberus, named after the monstrous three-headed guardian of the gates of the Underworld in Greek mythology. The planner incorporates three main recent improvements, namely enhanced mutex detection, recent novelty heuristic, and the extension of red-black planning heuristic to conditional effects. Two variants of the planner submitted to the International Planning Competition (IPC) 2018 differ in the red-black planning heuristic they use. In the remainder of this paper we describe the components in detail.

**Configurations**

Both Cerberus variants participate in three tracks, namely satisficing, agile, and bounded-cost. They are built on top of the adaptation of the Mercury planner (Katz and Hoffmann 2014a), runner-up of the sequential satisficing track of IPC 2014, to the recent version of the Fast Downward framework (Helmer 2006). Further, the implementation is extended to natively support conditional effects (Katz 2018). In contrast to Mercury planner, the red-black planning heuristic is enhanced by the novelty heuristic (Katz et al. 2017), replacing the queues ordered by the red-black planning heuristic $h_{RB}$ in Mercury planner with queues ordered by the novelty of a state with respect to its red-black planning heuristic estimate $h_{RB}$, with ties broken by $h_{RB}$. In what follows, we describe the parts that are shared between the tracks and then detail the configuration for each track.

**Enhanced Invariance Detection**

As the search and the heuristic computation are performed on the finite domain representation $SAS^+$ (Bäckström and Nebel 1995), invariance detection plays a significant role in the quality of the translation from PDDL representation to $SAS^+$. To reduce the number of multi-valued state variables we exploit the $h^2$ mutexes detection as a preprocessing step (Alcázar and Torralba 2015). In our preliminary experiments, this step was observed to make a significant contribution to the performance of the overall planning system.

**Red-Black Planning Heuristic**

In order to describe the configuration of the red-black planning heuristic $h_{RB}$, we need to specify how a red-black task is constructed (which variables are chosen to be red and which black), also known as painting strategy, as well as how the red-black task is solved. In both cases, we followed the choices made by Mercury planner. Specifically, for red-black task construction followed one of the basic strategies, namely ordering the variables by causal graph level, and either (a) iteratively painting variables red until the black causal graph becomes a DAG (Domshlak, Hoffmann, and Katz 2015), or (b) iteratively painting variables black as long as the black causal graph is a DAG. There are two submitted planners, that differ in their painting strategies. While the planner that (similarly to Mercury planner) uses strategy (a) is called Cerberus, the planner that uses strategy (b) is denoted by Cerberus-gl. These two planners differ in red-black planning task creation only, and therefore in what follows, we describe the configurations without mentioning the actual planner. The further difference from Mercury planner is in the definition of invertibility in the presence of conditional effects. In our planners we follow the definition of Katz (2018).

For solving the red-black task, we use the algorithm presented in Figure 2 of Katz (2018). It is an adaptation of the algorithm of Katz and Hoffmann (2014a) to tasks with conditional effects. The algorithm receives a red-black planning task, as well as a set of red facts that is sufficient for reaching the red-black goals. Such a set is typically obtained from a relaxed solution to the task. Then, it iteratively (i) selects an action that can achieve some previously unachieved fact from that set, (ii) achieves its preconditions, and (iii) applies the action. Finally, when all the facts in the set are achieved, it achieves the goal of the task. We follow Katz and Hoffmann (2014a) in the two optimizations applied to enhance red-black plan applicability: selecting the next action in (i) preferring actions such that achieving their black preconditions does not involve deleting facts from the set above, and selecting the sequences of actions in (ii), preferring those that are executable in the current state.

**Landmarks Count Heuristic**

Following the successful approaches of Mercury and LAMA planners, we use additional queues ordered by the landmark count heuristic (Richter and Westphal 2010).

**Novelty Heuristic**

The novelty heuristic used in our planners measures the novelty of a state with respect to its red-black planning heuristic estimate $h_{RB}$. Specifically, we use the $h_{QB}$ heuristic, as described in Equation 3 of Katz et al. (2017). The quantified both novel and non-novel heuristic $h_{QB}$ is designed not only to distinguish novel states from non-novel ones, but also to separate novel states, and even to separate non-novel ones. Consequently, we use the best performing overall configuration of Katz et al. (2017) in Cerberus planners.
Satisficing Track

The configuration runs a sequence of search iterations of decreasing level of greediness. The first iteration is the greedy best-first search (GBFS) with deferred heuristic evaluation, alternating between four queues. The first queue is ordered by the novelty of a state with respect to its red-black planning heuristic estimate $h^\text{RB}$, with ties broken by $h^\text{RB}$. The second queue consists of states achieved by preferred operators of the red-black planning heuristic $^\dagger h^\text{RB}$, ordered by $h^\text{RB}$. The third and forth queues are ordered by the landmark count heuristic, with all successors and those achieved by the preferred operators, respectively.

The next iterations perform a weighted $A^*$ with deferred heuristic evaluation and decreasing weights $w = 5, 3, 2, 1$, continuing with $w = 1$. All these iterations alternate between the four queues as in Mercury planner, with the first two ordered by $h^\text{RB}$, with all successors and those achieved by the preferred operators, respectively, and the last two as in the first iteration. In case a solution is found in the previous iteration, its cost is passed as a pruning bound to the next iteration.

In case of non-unit costs, a cost transformation is performed, adding a constant 1 to all costs. Further, the first iteration is performed twice, once with unit costs and once with the increased costs.

Agile Track

The configuration in the agile track mimics the first iteration of the configuration in the satisficing track as described above.

Bounded-Cost Track

The configuration in the bounded-cost track mimics the configuration in the agile track as described above. The only difference is that the cost bound is provided as an input.

References


---

$^\dagger$These are basically the preferred operators of the full delete relaxation, the FF heuristic.


