OLCFF: Online-Learning $h^{C\text{FF}}$

Maximilian Fickert and **Jörg Hoffmann**
Saarland Informatics Campus
Saarland University
Saarbrücken, Germany
{fickert,hoffmann}@cs.uni-saarland.de

Abstract

OLCFF is a sequential satisficing planner based on partial delete relaxation with explicit conjunctions using the $h^{C\text{FF}}$ heuristic. The heuristic can interpolate between fully delete relaxed semantics and real semantics by choosing the set of conjunctions C accordingly. Our planner is built around refining the heuristic online, which has proven to be the most effective way to use the $h^{C\text{FF}}$ heuristic. The main search algorithm used by the planner is a variant of enforced hill-climbing with online refinement and novelty pruning, followed by a LAMA-like anytime phase with GBFS and weighted A* where $h^{C\text{FF}}$ is used in a dual queue with a landmarks-count heuristic.

Introduction

In satisficing planning, heuristics based on the delete relaxation were part of most state-of-the-art planners for almost two decades (e.g. (Hoffmann and Nebel 2001; Richter and Westphal 2010; Katz and Hoffmann 2014)). However, the delete relaxation often ignores critical features of the planning task. These pitfalls can be diminished by "un-relaxing" part of the problem.

One approach to partial delete relaxation is red-black planning, where not all variables are relaxed, but only some of them (Domshlak, Hoffmann, and Katz 2015). The Mercury planner (Katz and Hoffmann 2014) is based on red-black planning, and was very successful at the last IPC.

We employ a different partial delete relaxation technique, where certain combinations of facts must be respected by the relaxed plans. For example in a transportation domain with fuel consumption, it can be useful to consider being in a specific location while still holding a certain amount oiif fuel. The $h^{C\text{FF}}$ heuristic implements this by treating a set of conjunctions C as atomic (Fickert, Hoffmann, and Steinmetz 2016). Choosing C correctly is critical for the performance of the heuristic, since, while the accuracy increases with larger C, so does the computational complexity.

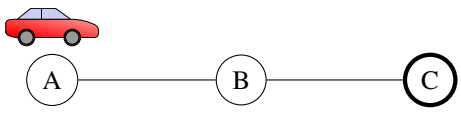
Fickert and Hoffmann (2017a) have recently shown that the $h^{C\text{FF}}$ heuristic is most effective when the conjunctions are generated online. They employ a variant of enforced hill-climbing (Hoffmann and Nebel 2001), called Refinement-HC, to detect when the search is stuck in a local minimum. Whenever this happens, the heuristic is refined by adding conjunctions to C until the local minimum is removed from the search space surface.

The OLCFF planner is based on online refinement with the $h^{C\text{FF}}$ heuristic. It uses an extension of Refinement-HC with novelty pruning (Lipovetzky and Geffner 2012) as its core search algorithm. The Refinement-HC phase is followed by a LAMA-like anytime search to find plans with better quality. This second phase runs $h^{C\text{FF}}$ in a dual queue with a landmarks-count heuristic (Richter, Helmert, and Westphal 2008).

$h^{C\text{FF}}$ and Refinement-HC

Delete relaxation heuristics can be made more accurate by taking *some* delete information into account. The $h^{C\text{FF}}$ heuristic generates partially relaxed plans that respect a given set of conjunctions C (Fickert, Hoffmann, and Steinmetz 2016). Achieving a conjunction $c \in C$ means achieving the individual facts represented by c *simultaneously*. Whenever a conjunction is a subset of the preconditions of an action, the conjunction of these facts must be achieved instead of the facts individually. Thus, e.g. if an action has two preconditions for which a conjunction exists, the partially relaxed plan must achieve both preconditions at the same time. A fully relaxed plan may achieve the first precondition, delete it again while achieving the second one, and can still apply the action.

Consider the task illustrated below. The car has to move from A to C. The car can only hold one unit of fuel, which each drive action consumes, but can be refueled at any location. Formally, there are STRIPS facts *at(x)* for the position of the car and *fuel* to indicate if the car has fuel. Initially the car is at location A and holds fuel.



A fully delete relaxed plan can ignore the fuel consumption and just apply the drive actions from A to B and B to C immediately after each other. The critical conjunction of facts that is ignored here is that the car must be at B while holding fuel before the second drive action can be executed. This conjunction can not be achieved by any of the drive actions as they consume the fuel fact. A partially re-

laxed plan generated by the $h^{C\text{FF}}$ heuristic respecting this conjunction would have to add the refuel action before driving from B to C, making the relaxed plan a real plan. In fact, with a sufficiently large set of conjunctions C, all plans generated by $h^{C\text{FF}}$ are real plans.

The $h^{C\text{FF}}$ heuristic works best when the conjunctions are generated online, in particular in Refinement-HC (Fickert and Hoffmann 2017a), which is an extension of enforced hill-climbing (EHC) (Hoffmann and Nebel 2001). Like standard EHC, the algorithm progresses through iterations of breadth-first search (BrFS) until a state s with lower heuristic value is found, then search continues from there. In Refinement-HC, these explorations are bounded by a fixed depth. If a state s with lower heuristic value can not be found within that bound, the heuristic is refined and the BrFS phase is restarted. Thus, Refinement-HC escapes local minima through heuristic refinement instead of brute-force search. A second extension to standard EHC are restarts from the initial state (without resetting the heuristic) whenever the search is stuck in a dead end. Due to the convergence of the partially relaxed plans generated by $h^{C\text{FF}}$ to real plans, Refinement-HC is complete.

Planner Components

The planner consists of two main search components. First, we run an extension of Refinement-HC with novelty pruning. The second search component runs GBFS with a dual-queue of the $h^{C\text{FF}}$ heuristic and a heuristic based on landmarks. This is followed by an anytime search with weighted A* using incrementally decreasing weights to obtain better plans (similar to LAMA).

Refinement-HC with Novelty Pruning

The core of our planner is an extension of Refinement-HC with novelty pruning (Fickert 2018). Instead of using BrFS with bounded depth in the local exploration phase, we perform exhaustive BrFS with incomplete novelty pruning, similar to a single iteration IW(k) of iterated width search (Lipovetzky and Geffner 2012). In our setting, a state passes the novelty test if it contains at least one novel *conjunction* $c \in C$, otherwise it is pruned. This corresponds to IW(1), but uses the conjunctions of $h^{C\text{FF}}$ instead of the individual facts. We denote this extension with conjunctions by IW(C). The novelty pruning is only applied in the BrFS phase, and thus only considers the states in the current BrFS exploration for pruning, not across the overall search. The simplified pseudo-code is shown in Algorithm 1.

This extension improves Refinement-HC as it takes the structure of the search space into account in the local explorations. Using novelty pruning here allows "interesting" branches (with novel states) to be explored in more depth. Sharing the set of conjunctions with $h^{C\text{FF}}$ also has a synergistic side-effect in Refinement-HC. The $h^{C\text{FF}}$ heuristic becomes more expensive to evaluate with each added conjunction, so refinement should be used carefully. On the other hand, IW(C) is less restrictive with each added conjunction. Thus, as Refinement-HC progresses, refinement will be triggered less frequently with larger C (since the novelty pruning is less aggressive), which reduces further overhead for $h^{C\text{FF}}$.

Algorithm 1: Refinement-HC with Novelty Pruning

```
s_best := I
while h(s_best) ≠ 0 do
    Run IW(C) from s_best until a state s with
      h(s) < h(s_best) is found.
    if no such state exists then
        Refine h in s_best.
        continue
    s_best := s
return SOLVED
```

GBFS and Weighted A*

Refinement-HC can not always overcome the limitations of local search. For example in Sokoban, the presence of deep dead ends has proven difficult, and global search algorithms like GBFS are much more suitable here.

Given these drawbacks, we place a time limit on Refinement-HC, as well as a growth bound on the number of conjunctions for the $h^{C\text{FF}}$ heuristic and run GBFS after Refinement-HC terminates. The growth bound on $h^{C\text{FF}}$ is motivated by the observation that in domains where Refinement-HC works well, the set of conjunctions typically does not grow excessively large.

The GBFS phase uses a dual-queue of $h^{C\text{FF}}$ and a landmarks-count heuristic (Porteous, Sebastia, and Hoffmann 2001; Richter, Helmert, and Westphal 2008). This makes it is very similar to LAMA (Richter and Westphal 2010), which uses a dual queue of h^{FF} and landmarks-count, but using $h^{C\text{FF}}$ in place of h^{FF}. The set of conjunctions for the $h^{C\text{FF}}$ heuristic is reset to a fixed size bound before starting the GBFS phase. During search, the heuristic then periodically replaces old conjunctions by newly generated ones, which allows it to adapt itself to the part of the search space that is currently being explored (Fickert and Hoffmann 2017b). Again similar to LAMA, when GBFS finds a solution, search continues with an anytime phase of weighted A* with incrementally lower weights. We cache heuristic values across the GBFS and weighted A* iterations to reduce overhead.

Implementation

The planner is implemented on top of Fast Downward (Helmert 2006). It performs a relevance analysis based on h^2 in the preprocessing phase (Alcázar and Torralba 2015), which simplifies the planning task by removing superfluous actions and facts. Additionally, it finds mutex relations between facts which are used by the $h^{C\text{FF}}$ heuristic to reduce its computational overhead.

In the competition, we build our planner with profile-guided optimization[1]. We generated profiling data by run-

[1] https://clang.llvm.org/docs/UsersManual.html#profile-guided-optimization

ning the planner on instances from previous IPCs. This data is then used to optimize the executable, e.g. by prioritizing the layout for frequently taken branches and making better inlining decisions.

Configurations

This section describes configuration details of our planner for the individual tracks.

Satisficing Track

The planner first runs Refinement-HC with unit action costs with a timeout of 1200 seconds, and bounds the complexity growth of the h^{CFF} heuristic to a factor of 8 compared to the heuristic without added conjunctions. If Refinement-HC finds a solution, it is restarted with original action costs. Otherwise, GBFS is run with unit action costs. Afterwards, we run GBFS and then weighted A* with weights 5, 3, 2, and 1, each with original action costs. In GBFS and weighted A*, h^{CFF} is used in a dual-queue with landmarks-count, but only uses the preferred operators of the h^{CFF} heuristic which improved results in preliminary experiments.

Agile and Cost-Bounded Tracks

In the agile and cost-bounded tracks, the planner also starts with Refinement-HC before running GBFS, but leaves out the anytime phase with weighted A* since we can stop after finding the first solution. In the agile track, the time limit for Refinement-HC is set to 180 seconds.

Acknowledgments. This work was partially supported by the German Research Foundation (DFG), under grant HO 2169/5-1, "Critically Constrained Planning via Partial Delete Relaxation".

References

Alcázar, V., and Torralba, Á. 2015. A reminder about the importance of computing and exploiting invariants in planning. In Brafman, R.; Domshlak, C.; Haslum, P.; and Zilberstein, S., eds., *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, 2–6. AAAI Press.

Clang compiler users manual, profile-guided optimization.

Domshlak, C.; Hoffmann, J.; and Katz, M. 2015. Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence* 221:73–114.

Fickert, M., and Hoffmann, J. 2017a. Complete local search: Boosting hill-climbing through online heuristic-function refinement. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17)*. AAAI Press.

Fickert, M., and Hoffmann, J. 2017b. Ranking conjunctions for partial delete relaxation heuristics in planning. In Fukunaga, A., and Kishimoto, A., eds., *Proceedings of the 10th Annual Symposium on Combinatorial Search (SOCS'17)*. AAAI Press.

Fickert, M.; Hoffmann, J.; and Steinmetz, M. 2016. Combining the delete relaxation with critical-path heuristics: A direct characterization. *Journal of Artificial Intelligence Research* 56(1):269–327.

Fickert, M. 2018. Making hill-climbing great again through online relaxation refinement and novelty pruning. In Bulitko, V., and Storandt, S., eds., *Proceedings of the 11th Annual Symposium on Combinatorial Search (SOCS'18)*. AAAI Press.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

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

Katz, M., and Hoffmann, J. 2014. Mercury planner: Pushing the limits of partial delete relaxation. In *IPC 2014 planner abstracts*, 43–47.

Lipovetzky, N., and Geffner, H. 2012. Width and serialization of classical planning problems. In Raedt, L. D., ed., *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI'12)*, 540–545. Montpellier, France: IOS Press.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the extraction, ordering, and usage of landmarks in planning. In Cesta, A., and Borrajo, D., eds., *Proceedings of the 6th European Conference on Planning (ECP'01)*, 37–48. Springer-Verlag.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39:127–177.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In Fox, D., and Gomes, C., eds., *Proceedings of the 23rd National Conference of the American Association for Artificial Intelligence (AAAI'08)*, 975–982. Chicago, Illinois, USA: AAAI Press.