Conformance Refinement for Online Hierarchical Planning, Diagnostics, and Plan Repair

PHD THESIS

Oliver Michael Kamperis, Marco Castellani and Yongjing Wang
School of Computer Science
University of Birmingham
United Kingdom

April 26, 2021

1 Introduction

This thesis proposes a novel approach to planning called *hierarchical conformance refinement*. The approach is intended to make complex discrete deterministic planning problems feasible to solve online. In conformance refinement, plans are generated over an abstraction hierarchy, containing multiple models of the planning domain/problem at different levels of abstraction. Unlike typical heuristic search methods, search is guided by a *conformance constraint addition* based approach. This requires that plans generated over the abstraction hierarchy achieve the same effects and remains structurally similar at all abstraction levels. This conceptually simple constraint provides the following major benefits:

1. *Problem Simplification* - Abstract models can reduce an exponential search space to linear, allowing abstract plans to be found exponentially faster that the concrete. The resulting abstract plan then guides search in the concrete model, by defining a *sequence of subgoal stages*, that the concrete plan must pass through, thus restricting the search space and reducing overall planning time exponentially, over directly solving the problem.
2. *Problem Division* - Abstract plans can be used to divide concrete planning problems into a sequence of distinct *partial problems*, only the first of which is solved prior to execution, reducing execution latency exponentially.
3. *Plan Repair* - In the real-world, plans are bound to fail. This occurs if an unexpected observation is obtained during execution. In conformance refinement, the hierarchical nature of generated plans can be used to identify which specific parts have failed, and which parts can be re-used. This allows the planner to save time, by regenerating only the failed parts of a plan. The downside, is that repaired plans become progressively worse.

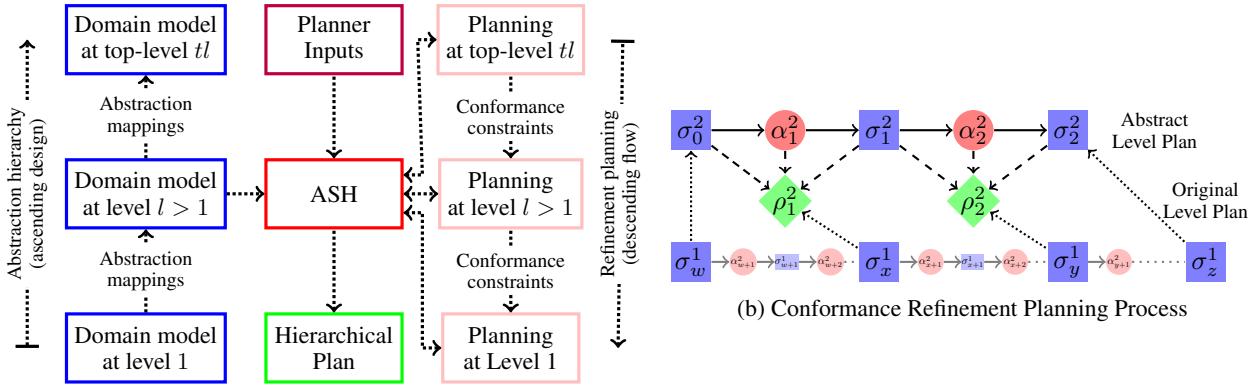
The trade-off of conformance refinement is that plans are not guaranteed to be optimal in the classical sense. It is hard to know if the speed benefit provided by conformance refinement will justify any loss in plan optimality ahead of solving.

Creation of abstract models of a planning domain is fundamental to the approach. This involves deciding; which elements of a domain are details that can be abstracted away, or what assumptions can be made to simplify search. The abstraction hierarchies used in refinement are simple to obtain, even for complex problems, with amorphous structure.

1.1 Structure and Operation of ASH the Conformance Refinement Planner

The proposed planner is called ASH. The diagram in Figure 1a shows the abstract structure of ASH. Its parts, their functions/dependencies, and control/information flow between them (as indicated by the arrows) are as follows:

1. The central red box is the ASH planner itself, a non-monotonic logic program containing only ≈ 50 rules.
2. The left-hand blue boxes represent a planning domain definition, an arbitrary size hierarchy of domain models.
3. The top purple box are the problem-specific inputs to the planner, e.g. initial state and final goals.
4. The right-hand pink boxes indicate the flow of hierarchical conformance refinement planning, plan at each level in descending order, passing conformance constraints generated at the previous level to the next.
5. The bottom green box is the resulting hierarchical plan, a sequence of *refinement trees*, data structures used to represent conformance refinements, defining a hierarchy of conforming classical plans over all abstractions.



(a) Structure of ASH

(b) Conformance Refinement Planning Process

Figure 1: Conformance Refinement Planning with ASH

The diagram in Figure 1b depicts the conformance refinement process. The blue squares are states, red circles actions, and green diamonds subgoal stages. The upwards dotted arrows indicate states and subgoal stages that conform with (map to) each other. The downwards dashed arrows are the creation of subgoal stages from abstract state transitions.

The top-level is a plan generated in an abstract domain model and the bottom-level is the refinement of that plan generated in the original model (the model from which the abstract was created). Each abstract state transition creates one subgoal stage, forming the conformance constraint for the next level. The plan at the next level must both satisfy any reintroduced planning constraints of the original model (that were removed in the abstract) and must achieve these subgoal stages (i.e. satisfy the conformance constraint) by passing through states that satisfy them in the same order. This restricts the search space, guiding search by allowing only plans the satisfy it, thus making it easier to find a plan. Each subgoal stage can be seen as being achieved by a distinct partial plan. The start state of each non-initial partial plan is known only to satisfy the previous subgoal stage, and is otherwise unknown until that stage has been achieved by the previous partial plan. The initial state and final goal at each level is fixed and must conform with each other.

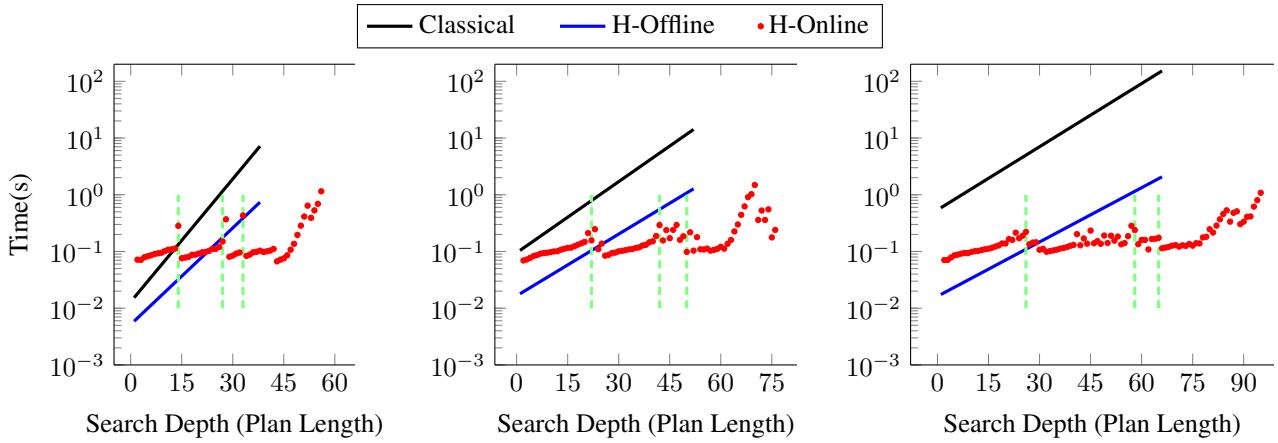
2 Current Progress and Future Work

Thus far, the following parts of ASH and the thesis have been completed and written up.

- **Implementation and testing of planning algorithms:** The refinement planning part of ASH is now complete. The system currently provides support for two types of abstract domain model. Two other types of model have been implemented in an older version of ASH, these will be updated and formalised as time permits.
- **Experimentation of planning:** We have performed experimental tests for the planning algorithms on an extension of the classic blocks world domain. The results (given below) have been very promising thus far.
- **Partial implementation of diagnostics system and multiple robot systems:** The diagnostics and multi-robot systems have also been implemented in an older version of ASH, but need to be updated and adapted to allow for experimentation. In particular, the system needs an external program to feed it simulated observations.

The following are the objectives which we plan to achieve during the rest of this thesis.

- **Implementation and formalisation of plan repair systems:** This system is needed to process obtained observations, identify the failed part of a hierarchical plan, call the diagnostic system to resolve any contradictions in the current state, call the planner to regenerate the failed partial plan, and finally process the plans to form a repaired complete plan. If any goal is made unachieveable by an observation, ASH will attempt to achieve the highest utility sub-set of goals. The theory for this is simple, the implementation is likely to be complex.
- **Integration with Bluebear:** This will be done to perform experimentation of diagnostics, plan repair, and multiple robot systems, on a more complex robotic supermarket domain including a heterogeneous robot team.
- **Disassembly sequence planning for electric vehicle lithium-ion batteries ase study:** This is a challenging problem which is receiving a lot of attention in research. It is a problem to which refinement planning approaches have not before been applied to and it also presents all the problems that ASH is designed to overcome, making is an excellent case study choice. To do this, a bespoke type of abstract model will be designed, that identifies sub-assemblies to be disassembled first, before the more complex components are.



(a) Solution Checking Time: Problem 1 (b) Solution Checking Time: Problem 2 (c) Solution Checking Time: Problem 3

Figure 2: Solution Checking Time respective of Plan Length and Execution Latency

2.1 New Results

Three different problem instances of increasing complexity were run on an extension of the classic blocks world planning problem. Experiments were run for each of; classical, hierarchical offline (h-offline) (offline uses no problem division), and hierarchical online (h-online) planning. For h-online, the problems were divided into four partial problems.

A small sub-set of our results are given in Figure 2. These give the solution checking times respective of plan length. The solution checking time is the time taken to exhaust the search space and determine if a plan exists, for each step up to the given plan length. It is thus a measure of the complexity of the search space, relative to plan length. For classical and h-offline, exponential trend lines are fitted to the available data. Whereas, for h-online, the data is plotted directly, since no clear global exponential trend exists. The vertical dashed green lines indicate the average time step upon which a partial planning problem completed and moved to the next, i.e. they are the problem division points.

The results indicate that; offline refinement planning consistently reduces total planning time exponentially over classical planning in a diverging trend, and online refinement planning yields partial plans exponentially faster than complete planning. This allows a robot to begin execution in a fraction of the time taken to generate a complete plan via classical planning. The benefit is made stark in consideration that classical takes on average 1692.6s to solve our most complex test problem, whereas offline refinement takes 98.8s, and online refinement just 21.3s to generate the complete plan and 5.56s to generate the first partial plan, resulting in a total of a 99.68% reduction in execution latency.

However, the approach still has important facets, which we have categorised and summarise below:

1. Decisions can be made which are optimal in abstract problems, but are poor with respect to the original problem. During refinement, these decisions must be maintained by conformance, and can lead to poor plans.
2. In online planning, decisions can be made during earlier partial problems which are locally optimal, but not globally optimal respective of the complete problem and final goal. This often leads to worse long-term plans.

2.2 Plan of Future Work

My extension request has been accepted and my registration dates have changed. The end of minimum period of registration has been extended from 30/09/2021 to 30/12/2021 and deadline for thesis from 30/09/2022 to 30/12/2022.

The current plan of future work up until the end of the thesis is shown in Figure 3. Following the RSMG 6 we will work towards implementing the diagnostic and plan-repair systems into ASH. This will result in a complete version of ASH which is fully capable of online planning, diagnostics, and plan repair, as well as integration with Bluebear for experimental testing. This work will be written up into a publishable paper and presented in the RSMG 7 meeting in November 2021. Unfortunately, at this point is does not seem feasible to perform experimental testing of ASH against other general domain-independent planning paradigms. Following the RSMG 7, we will then develop an ASH compatible model for disassembly sequence planning problems. It is however, not yet clear what the nature and complexity of the disassembly problem we will tackle will be. We will then use the extra time granted by the extension primarily to focus on writing up the disassembly sequence planning into a third paper. We hope to have sufficient time to evaluate the effectiveness of our approach against other existing approaches to disassembly sequence planning.

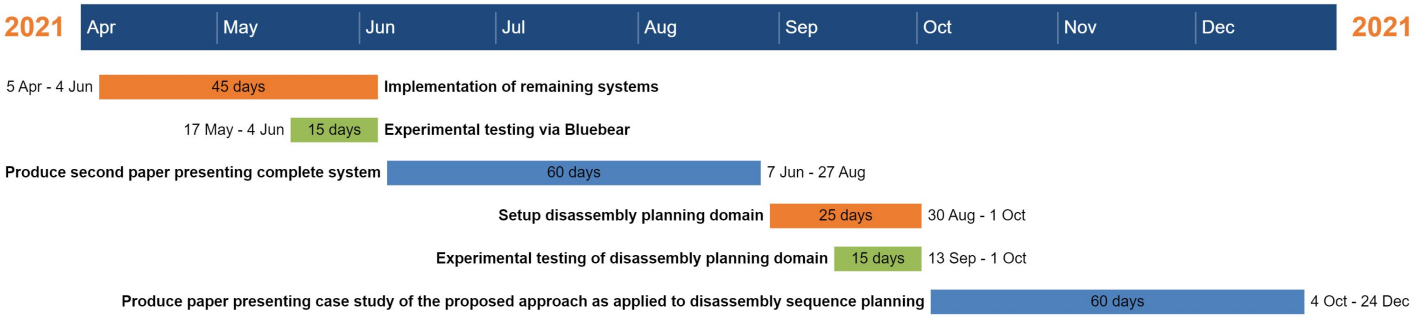


Figure 3: Plan of future work up until the end of year 3.