A Multi-Language Heuristic Driven HTN Planner in Python

Project Plan

Cael Milne

u34cm18@abdn.ac.uk

Department of Computing Science,
University of Aberdeen, Aberdeen AB24 3UE, UK

Introduction

AI planning is an exciting world with applications in a variety of fields i.e. robotics, travel planning, etc. Hierarchical Task Network (HTN) Planning is an extension to traditional planning which makes use of hierarchies of tasks known as task networks. Tasks can either be primitive tasks or compound tasks - which need to be split into several primitive tasks. HTN planners require problem environments to be well-defined, which makes HTN planners faster and more scalable than traditional planners [1].

HTN planning was recently introduced as a challenge at the International Planning Competition in 2020 [2]. The challenge made use of the HDDL input language which is based upon the PDDL and PANDA input languages. HDDL offers functionality for all regular HTN planning concepts such as task networks, planning problems and domains, as well as subtasks, task ordering, and preconditions. HDDL is proposed as a common input language of HTN planning [3].

Goals

My aim for this project is to develop a HTN planner in Python capable of solving HDDL and SHOP problems. HTN problems can be defined as total ordered or partial ordered, this project sets out to be compatible with both.

Parsers for HDDL and SHOP already exist in several languages. These can be used as guidance when implementing similar parsers in Python. For the case of developing the HDDL parser, I have been given access to a PDDL parser written in Python as a starting point. Converting both input languages into some data form that allows processing by the algorithm is the central task.

With regard to the actual solving algorithm, there is a range of algorithms already developed and used by other planners. But for the purposes of this project I intend on building a new one based on some concepts discussed by Daniel Holler, Pascal Bercher, Gregor Behnke, and Susanne Biundo [4]. These concepts talk about ways to guide the algorithm effectively towards a solution.

As the field of HTN planning progresses, inevitably new input languages will be introduced. It is key that the system developed for this project maintains this thought in its software design. Good design practices - such as the hexagonal pattern - make it possible for future languages and modifications to be brought forward relatively easy. The first test of the quality of software engineering will be during the development of support for the second input language in this project.

Similarly, when developing support for heuristics, an implementation similar to the PANDA planner - which supports multiple heuristics - would be ideal. The interchangeability of heuristics can be easily trialled by simply developing a testing multiple heuristic functions with the planner.

Python was selected at the language of choice for this project since there is a lack of planners developed in Python. Planners are commonly written in C++ and Java. As well as the language of choice, another concern is for future users who want to modify the planner and try their own ideas. This project aims to allow modifications to input, processing, and planning seamlessly to aid future research in HTN planning.

If I am able to implement a working system that satisfies all the goals laid out, some further features can be added. The first additional feature would be the inclusion of the output format required in the IPC. Which requires a description of the decomposition's done during search to be returned at output as well as the resulting plan. On top of this additional and more complex search heuristics can be researched and added to the system.

Methodology

The general activities regarding this project are:

- Researching related work. To gain a deeper understanding of planners and input languages.
- Learning about current planners and how they work
- Testing code prototypes to determine what is successful and what needs to be avoided
- Testing code written with autonomous test cases

Relevant Work

Some existing planners satisfy some goals of this project already. The PANDA planner can operate with a selection of interchangeable heuristics. Support for partially ordered problems is provided by the PANDA planner. In its current version PANDA only supports one input language - HDDL [5].

The HyperTensioN system is split into two parts. The first section containing the parsers, middle-end and, compilers in a module called *Hype*. The second section is the

HyperTensioN module which solves the planning problem with the Ruby compiler. HyperTensioN provides functionality for multiple input languages and is built with support for new languages in mind. The current languages supported are PDDL, JSHOP and, HDDL [6].

Resources Required

The resources for this project will be minimal since its purely software based. Hence, I will require a PC which is able to run Python, and a Python interpreter. Other necessary resources are purely academical such as information on existing systems.

Some software resources are also required. GitHub for version control and an IDE from developing in Python (PyCharm or VS Code), are essential to this project.

Risk Assessment

The goals mentioned above are time-bound to the deadline of this project (May 20th 2022). As such some risks exist, mitigating factors for the risks involved are described in this section.

A potential issue is not able to get system to work with both HDDL and SHOP. A method of mitigating this could be to re-base the code written for HyperTensioN [6] from Ruby to Python.

It is hoped that the developed system will be able to work with both total ordered and partial ordered problems. The functionality for partial ordered problems could prove too strenuous given the time limitations, to combat this support for partial ordered problems would be abandoned.

Implementing a good heuristic to improve the efficiency of the planning search could again prove outside the time constraints of this project. As such the heuristic used would be scaled down to a more basic level.

Any issues implementing the planning algorithm will be overcome by studying and adapting features found in other planning algorithms from other planners to suit the aims of this project.

The threat of technical issues impacting the code base or report will be mitigated by keeping regular backups locally as well as ensuring cloud storage is kept up to date. The code base will be stored securely on GitHub and the report will be kept on Overleaf.

Timetable

The main categories of activities required for this project are the report write-up and software implementation. The writing of the report will take place mostly in the opening and closing weeks of term. Meanwhile, software implementation is scheduled to take place over the middle weeks of the term as shown in Figure 1.

The project sets off with the first three sections of the report, Introduction, Background & Related work, and Research Question & Requirements. These three sections are before implementation since the knowledge and work required in these sections is necessary for the system development.

The implementation of the system is split into distinct sections. The first section aims to create a HDDL parsing component which works with totally ordered problems. Then a basic solving algorithm will be added, so the system can begin to be tested thoroughly. After testing; the SHOP parser can be added and tested. Once both input languages are accepted by the system the solving algorithm will be upgraded with a heuristic. Functionality for partial ordered problems will be the last component implemented before commencing evaluation of the system. To evaluate the system, existing planners will be compared to the one developed.

After coding has concluded, the remaining sections of the report will become the centre of attention. A write-up of the implementation -with help from a development diary- will be first. Following the implementation write up, a week is set aside for evaluation of the system. Evaluation write up will coincide with the code interaction for comparing systems. Finally, the conclusion and discussion section of the report will summarise all aspects of this project.

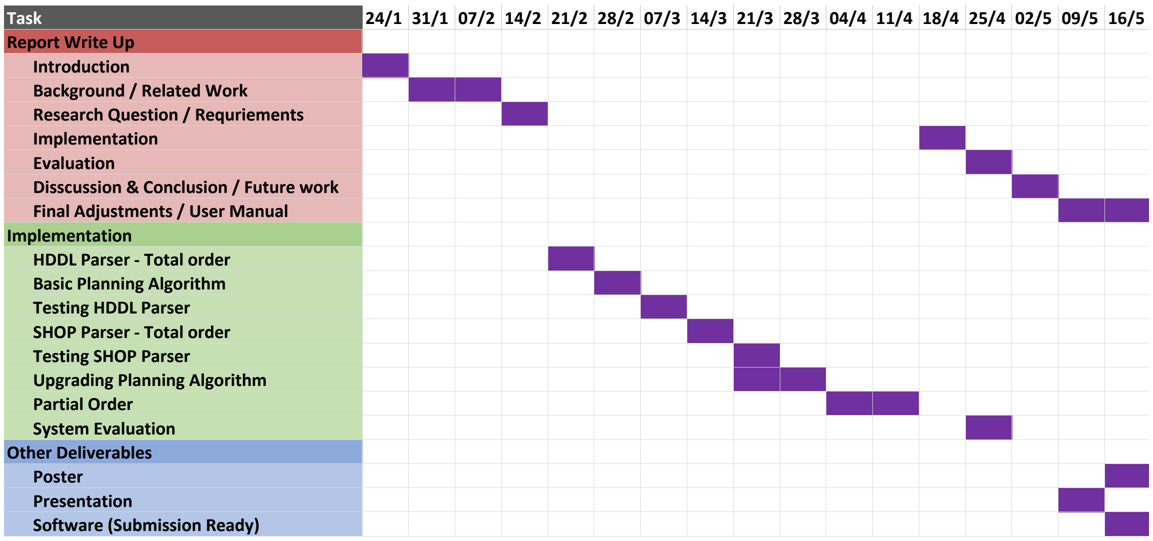
I have left a couple of weeks at the end of term to allow for thorough writing up of the report which will include creating an abstract. The user manual for the code base will also be finalised in this period.

The deadline set for the project is the 20th of May. As such it is hoped that the software will be completed a couple of weeks beforehand to account for the preparation and execution of the live demo as well as the poster.

Deadline	Task
31/1	Report Introduction
14/2	Background and Related Work Write up
21/2	Research Question and Requirements
14/3	System supports HDDL and has basic solver
28/3	System supports SHOP problems
04/4	System uses Heuristic
18/4	System supports Partial ordered problems
25/4	Implementation Write up
2/5	System evaluation
9/5	Conclusion
16/5	Live Demo Presentation
20/5	Project Deadline - Submission ready

Table 1: Milestone Chart

Appendix



Task	24/1	31/1	07/2	14/2	21/2	28/2	07/3	14/3	21/3	28/3	04/4	11/4	18/4	25/4	02/5	09/5	16/5
Report Write Up																	
Introduction	■																
Background / Related Work		■	■														
Research Question / Requriements				■													
Implementation													■				
Evaluation														■			
Disscussion & Conclusion / Future work															■		
Final Adjustments / User Manual																■	■
Implementation																	
HDDL Parser - Total order					■												
Basic Planning Algorithm						■											
Testing HDDL Parser							■										
SHOP Parser - Total order								■									
Testing SHOP Parser									■								
Upgrading Planning Algorithm									■	■							
Partial Order											■	■					
System Evaluation														■			
Other Deliverables																	
Poster																	■
Presentation																■	
Software (Submission Ready)																	■

Figure 1: Main Project Activities

References

[1] Ilce Georgievski and Marco Aiello. An overview of hierarchical task network planning. *CoRR*, abs/1403.7426, 2014.

[2] G Behnke, D Höller, P Bercher, S Biundo, Damien Pellier, H Fiorino, and R Alford. Hierarchical Planning in the IPC. In *Workshop on HTN Planning (ICAPS)*, Berkeley, United States, July 2019.

[3] Daniel Höller, Gregor Behnke, Pascal Bercher, Susanne Biundo, Humbert Fiorino, Damien Pellier, and Ron Alford. Hddl: An extension to pddl for expressing hierarchical planning problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9883–9891, 2020.

[4] Daniel Höller, Pascal Bercher, Gregor Behnke, and Susanne Biundo. A generic method to guide htn progression search with classical heuristics. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*, 2018.

[5] Daniel Höller, Gregor Behnke, Pascal Bercher, and Susanne Biundo. The panda framework for hierarchical planning. *KI-Künstliche Intelligenz*, pages 1–6, 2021.

[6] Maurício Cecílio Magnaguagno, Felipe Meneguzzi, and Lavindra de Silva. HyperTensioN: A three-stage compiler for planning. In *Proceedings of 10th International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track (IPC 2020)*, pages 5–8, 2021.