D4.3: Repository of the Planning Engines

June 30th 2022

Project funded by the European Commission within the Horizon 2020 Programme		
Dissemination Level		
PU	Public	X
CO	Confidential, only for members of the consortium (including the Commission Services)	□
CL	Classified, as referred to in Commission decision 2001/844/EC	□

Deliverable number:	**D4.3**
Deliverable name:	Repository of the Planning Engines
Work package:	WP4: Planning Engines
Lead WP:	UNIBS
Lead Task:	UNIBAS, UNIBS, FBK, UNIROMA1, CNRS, ORU

Contents

Document Revision History

Date	Issue	Author/Editor/Contributor	Summary of main change
06/06/2022	V1	Andrea Micheli (FBK)	Initial structure and introduction
20/06/2022	V2	All WP4 Partners	Contributions to each task section
25/06/2022	V3	All WP4 Partners	Final proofreading

Abstract

Automated Planning and Scheduling is a central research area in AI that has been studied since the inception of the field and where European research has been making strong contributions over decades. Planning is a decision making technology that consists in reasoning on a predictive model of a system being controlled and deciding how and when to act in order to achieve a desired objective. It is a relevant technology for many application areas that need quick, automated and optimal decisions, like agile manufacturing, agrifood or logistics. Although there is a wealth of techniques that are mature in terms of science and systems, several obstacles hinder their adoption, thus preventing them from making the footprint on European industry that they should make. For example, it is hard for practitioners to find the right techniques for a given planning problem, there are no shared standards to use them, and there is no easy access to expertise on how to encode domain knowledge into a planner.

The AIPlan4EU project will bring AI planning as a first-class citizen in the European AI On-Demand (AI4EU) Platform by developing a uniform, user-centered framework to access the existing planning technology and by devising concrete guidelines for innovators and practitioners on how to use this technology. To do so, we will consider use-cases from diverse application areas that will drive the design and the development of the framework, and include several available planning systems as engines that can be selected to solve practical problems. We will develop a general and planner-agnostic API that will both be served by the AI4EU platform and be available as a resource to be integrated into the users' systems. The framework will be validated on use-cases both from within the consortium and recruited by means of cascade funding; moreover, standard interfaces between the framework and common industrial technologies will be developed and made available.

Executive summary

In the AIPlan4EU project, we are developing an abstraction layer, called Unified Planning Framework (UPF), which abstracts away the specificities of planning engines that are being integrated in the context of Work Package 4. In this document, we report the current development status of all the planning engines being developed within WP4.

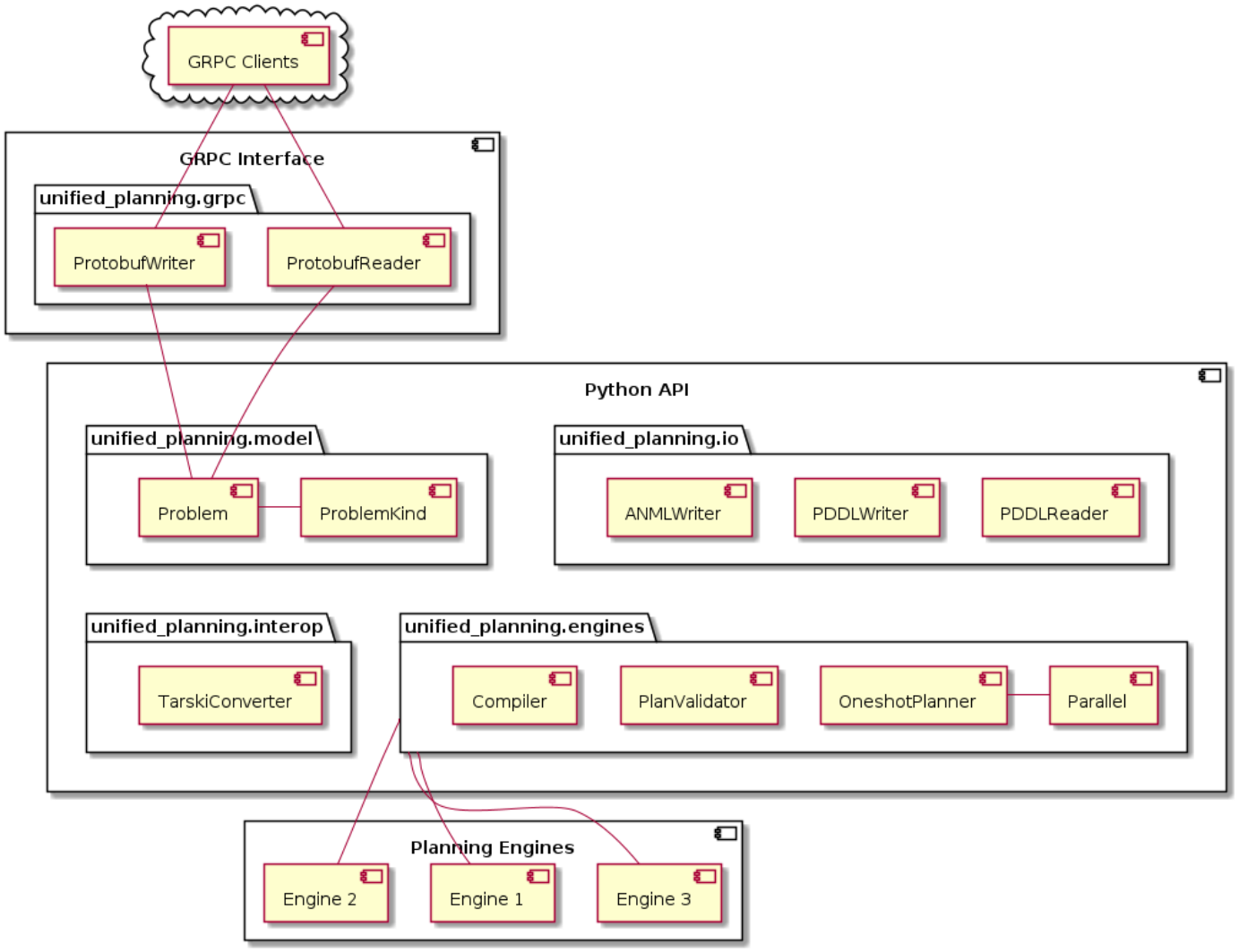
In particular, we created a repository for each planning engine, because the UP library developed within WP3 offers a plugin system. Each engine wrapper is therefore developed externally to the central UPF repository, allowing anyone to easily contribute more engines. Nonetheless, all the repositories developed within the project are summarized under the same “organization” in GitHub.

In this document, we report, for each of the integrated planning engines, the specific characteristics of the integration as well as a description of the integration status and the foreseen next steps .

Introduction

The unified Planning Framework developed by the AIPlan4Eu project within WP3, is incarnated as an open-source Python3 library (called Unified Planning Library - UP Library for short) that offers a unified API for the specification and manipulation of planning problems and integrates a number of planning engines, developed and integrated within WP4, that offer planning technology services.

The general architecture of the UP is shown in the following picture.



The UP Library is composed of four main packages and offers both a python API and a grpc interface. The library provides the functionality needed to model and to transform planning problems and organizes the possible queries to the planning engines in a series of Operation Modes. An operation mode defines how the user can interact with a certain planning engine in an abstract way: each planning engine must define which is the subset of operation modes that it is able to support. In this way, the library is capable of automatically selecting a planning engine for a certain operational mode taking into account the specific characteristics of the problem formulated by the user.
The planning engines themselves are not included in the UP Library, but it is possible to add planning engines externally by means of a plugin system offered by the `unified_planning.engines` package. Moreover, the installation system of the UP library allows the automated installation of several commonly used engines.

The UP library defines the set of Operation Modes as a set of interfaces to be instantiated by each engine.

Within work package 4, we decided to develop the integration of each of the planning engines contributed by the partners participating in the work package in a separate repository. The development of each of such integration is publicly visible in the AIPlan4EU GitHub organization[1] and all the integrations are developed as open source software (note, however, that the planning engine being integrated might or might not be open source and the license is reported both in this document and in each repository "readme" file).

In this document, we report the current status of the development of the planning engines within work package 4. In particular, we provide links to the external repositories where the development of each planning engine is happening and we provide an overview of the capabilities of the different engines currently being integrated in the unified planning framework.

Planning Engine Repositories

The subsequent sections are organized according to the task structure of work package 4. For each task in the work package, we list the planning engines integrated within that task following a common structure. In particular, we provide the link to the repository where the integration code is, the license specification, the constraints on the platform where the engine can be used (if any), the list of operation modes supported by the planning engine and a general description of the planning and in itself. Note also that several engines also provide a simplified installation method using the Python Package Index PyPI[2] (See for example the demo notebook we recently presented at the ICAPS conference: https://bit.ly/UPDemo).

Task 4.1 Classical Planning

Two classical planners have been integrated so far, namely pyperplan and fast-downward. In addition to these, also all the other planners that will be presented in other tasks support classical planning as a special case (obviously, classical planners are usually more efficient than more general plan generation techniques).

Pyperplan

Repository: https://github.com/aiplan4eu/up-pyperplan
License: GPL-3.0
Platforms: Any
Operation Modes: one-shot planning
Description: Pyperplan[3] is a lightweight STRIPS planner written in Python. It is integrated in the UP without using intermediate formal languages, that is we directly construct the pyperplan data structures from the UP ones. The pyperplan code is deliberately written to prefer readability and clarity over efficiency, and thus it has been used as a prototypical planner from the early phases of the project. Pyperplan supports very basic classical planning problems, without disjunctive or negative preconditions and without conditional effects. Despite these limitation, the integration of pyperplan is very efficient. Finally, pyperplan and its integration are platform-independent as the whole planner is written in python3.

Available on Pypi at https://pypi.org/project/up-pyperplan
Installation: `pip install up-pyperplan`

[1] https://github.com/aiplan4eu
[2] https://pypi.org/
[3] https://github.com/aibasel/pyperplan

Fast Downward

Repository: https://github.com/aiplan4eu/up-fast-downward
License: GPL-3.0
Platforms: Linux, MacOS, Windows
Operation Modes: one-shot planning
Description: Fast Downward[4] is a well-established system for classical planning and covers a wide range of heuristic search approaches. It supports all features of classical planning as defined in the classical fragment of PDDL. Fast Downward is implemented in Python and C++ with an emphasis on the efficiency of the planning system. To make it available on all platforms, the UP integration supports the compilation on all major platforms and provides precompiled versions as wheels via the Python package index PyPI.

Available on Pypi at https://pypi.org/project/up-fast-downward
Installation: `pip install up-fast-downward`

Task 4.2 Numeric Planning

ENHSP

Repository: https://github.com/aiplan4eu/up-enhsp
License: GPL-3.0
Platforms: Any
Operation Modes: one-shot planning
Description: ENHSP is an expressive numeric planner supporting planning problems involving Boolean and numeric state variables, actions, processes and events as those that can be expressed using the PDDL+ language. A distinctive feature of ENHSP is the ability to reason over problems with a prevalent numeric structure, which may involve linear and non-linear numeric conditions.

Available on Pypi at https://pypi.org/project/up-enhsp
Installation: `pip install up-enhsp`

LPG

Repository: https://github.com/aiplan4eu/up-lpg
License:
Platforms: Ubuntu 20.04, Windows x64, MacOs
Operation Modes: one-shot planning
Description: LPG is an automated planner for classical planning, numeric planning, and temporal planning. It supports propositional variables, numeric state variables, timed-initial literals, derived predicates, and durative actions, as in PDDL2.2. LPG can operate under different modalities: one-shot satisficing planning, incremental planning, and plan repair. The planner is written in C, and it can be run in either speed mode or quality mode, in order to quickly find a solution (without considering its quality) or search for good-quality solutions, respectively. The planner exploits a particular representation based on action graphs forming a search space that is explored by stochastic local search procedures using heuristics based on relaxed plans.

[4] https://www.fast-downward.org/

Task 4.3 Temporal Planning

Within task 4.3, Tamer, a planner developed at FBK, has been integrated for being used in the UP Library. In addition to Tamer, also the previously discussed LPG planner is capable of handling temporal domains. Moreover, the Aries planner, which will be discussed in task 4.5, and the Spiderplan tool described in task 4.6 are capable of temporal reasoning. So, in total we can count on 4 temporal planners currently integrated within the UP.

Tamer

Repository: https://github.com/aiplan4eu/up-tamer
License: Proprietary but free for non-commercial use
Platforms: Windows, MacOS and Linux on X86_64 hardware
Operation Modes: one-shot planning, validation
Description: TAMER (https://tamer.fbk.eu) is an application-oriented planner for the ANML (read as "animal") planning specification language. The objective of TAMER is to provide functionalities to model, solve and analyze expressive temporal planning problems in practice. Moreover, Tamer is capable of simulating temporal planning problems and provides a range of different algorithms using satisfiability modulo theories and heuristic search. Tamer offers its functionalities both via a command-line interface and through an API that is usable from the major programming languages, including Python, Java and C++.

Available on Pypi at https://pypi.org/project/up-tamer
Installation: `pip install up-tamer`

Task 4.4 Multi-Agent Planning

FMAP

Repository: https://github.com/aiplan4eu/up-fmap
License: GPL 3
Platforms: Any (MacOSX and Windows to be tested)
Operation Modes: one-shot planning
Description: FMAP is a cross-platform Java-based software for Multi-Agent Planning (MAP) in non-deterministic environments, where agents have a local view of the world. FMAP uses a distributed heuristic search strategy, where each planning agent features an embedded search engine based on a forward partial-order planning scheme, which allows the agents to plan their actions in parallel whenever possible, thus largely improving the quality of solution plans. The FMAP platform makes extensive use of Java *interfaces*, which encapsulate the different components of the code, enabling for an easy replacement of the existing algorithms. This feature greatly simplifies the arduous task of developing and integrating new distributed planning algorithms and heuristic functions. Additionally, FMAP includes a thoroughly-tested communication infrastructure that allows for the exchange of synchronous and asynchronous messages.

Task 4.5 Refinement Planning

Aries

Repository: https://github.com/aiplan4eu/up-aries
License: MIT

Platforms: Linux, Windows, MacOS
Operation Modes: one-shot planning
Description: Aries is an automated planner targeting hierarchical and temporal problems. The objective of Aries is to model and solve hierarchical problems with advanced temporal features and optimization metrics. It relies on the proximity of these with scheduling problems to propose a compilation into a constraint satisfaction formalism. Solving exploits a custom combinatorial solver that leverages the concept of optional variables in scheduling solvers as well as the clause learning mechanisms of SAT solvers.
Integration into the AIPlan4EU project is ongoing, synchronized with the effort for augmenting the modeling capabilities of the unified-planning library to model hierarchical problems.

Task 4.6 Task and Motion Planning

Spiderplan

Repository: https://github.com/aiplan4eu/up-spiderplan
License: GPL 3
Platforms: Windows, Linux
Operation Modes: one-shot planning
Description: Spiderplan is a constraint-based automated planner based on flaw resolution. As a baseline it includes support for solving goals, temporal constraints, reusable resources, general constraint processing, and interaction constraints for human-aware planning. One of the main features of Spiderplan is that it can be extended with new types of constraints. One such extension is the integration with motion planning through the coordination_oru framework.

The integration with the unified planning framework is ongoing. The current focus is the translation of the unified planning API to the constraint databases used by Spiderplan.

Conclusion

In this document we summarized the status of the development of the integration of the planning engine within work package 4. In addition to the current development carried on by the project partners, in a few months there will be additional integrations provided by external supporters recruited by means of the cascade funding programme of the project. Therefore, we expect to significantly enrich the set of planning engines available from the unified planning framework both by covering additional areas of planning and by augmenting the engines in the existing tasks. The final results of the integration work will be reported in Deliverable D4.1.