PLANUTILS: Bringing Planning to the Masses

Christian Muise,[1] Florian Pommerening,[2] Jendrik Seipp,[3] Michael Katz [4]

[1]Queen's University
[2]University of Basel
[3]Linköping University
[4]IBM T.J. Watson Research Center
christian.muise@queensu.ca, florian.pommerening@unibas.ch, jendrik.seipp@liu.se, Michael.Katz1@ibm.com

Abstract

PLANUTILS is a general library for setting up Linux-based environments for developing, running, and evaluating planners. Over the last decades, the planning community has produced countless solvers for various planning formalisms, as well as many other tools to help the planning practitioner. From state-of-the-art planners, over validators, to parsing libraries, the planning ecosystem has grown quite large. In the demo, we highlight an effort that aims to unify this ecosystem and make it seamless for users to get started with what the ICAPS community has to offer.

Demo: mulab.ai/demo/planutils

1 Motivation

Many AI research fields are moving towards a model of shared infrastructure to interface with modern research software. The growing ecosystem of planning-based tools and solvers is, however, currently hosted on personal or lab spaces online, with tools often requiring a non-trivial setup effort. PLANUTILS fills this gap within the planning community. The project offers friction-free access to a wide range of existing planners, and the scope of functionality continues to grow. Additionally, various modes of interacting with this growing functionality are under active development so that working with any of the modern planning systems is accessible to everyone.

2 Packages

PLANUTILS is based on a *package* system, similar to PyPI (Python Software Foundation 2022) and Anaconda (Anaconda Inc. 2022). In this section, we describe how to add a new package (e.g., for a new planner) to PLANUTILS, and show some of the packages that are already included.

2.1 Configuration

Configuring a new package requires four elements, each provided here as an example. First, a JSON manifest is required to describe the package (including its size once installed):

```
{"name": "Fast␣Downward",
 "description": "fast-downward.org",
 "install-size": "36M",
 "dependencies": []}
```

The next element is a script for installing the package. Typically this is as simple as fetching a pre-compiled image for the planner (the PLANUTILS documentation includes instructions for building images):

```
#!/bin/bash
singularity pull --name \
 downward.sif shub://aibasel/downward
```

Similarly, we require a script to remove a package:

```
#!/bin/bash
rm downward.sif
```

And, finally, a script for running the package/planner (note that all command-line arguments are passed through):

```
#!/bin/bash
singularity run -e \
 $(dirname $0)/downward.sif $@
```

By defining dependencies between packages, one can share common functionality. For example, the `lama` package has empty install/uninstall scripts, and depends on the `downward` package:

```
{"name": "LAMA",
 "description": "fast-downward.org",
 "install-size": "20K",
 "dependencies": ["downward"]}
```

```
#!/bin/bash
planutils run downward -- \
  --alias lama $@
```

2.2 Library Scope

The initial offering of PLANUTILS contains packages for a wide variety of planners and planning utilities. Full details can be found in the PLANUTILS command-line help. In summary, it includes access to the following utilities and planners (along with several variations): Cerberus (Katz 2018), Delfi (Katz et al. 2018a), Fast Downward (Helmert 2006), ENHSP (Scala et al. 2017), ForbidIterative (Katz et al. 2018b; Katz and Sohrabi 2020; Katz, Sohrabi, and Udrea 2020), hpddl2pddl (Alford, Kuter, and Nau 2009), K* (Katz

et al. 2018b), OPTIC (Benton, Coles, and Coles 2012), planning.domains (Muise 2016a), Pyperplan (Alkhazraji et al. 2020), Scorpion (Seipp, Keller, and Helmert 2020), SMT-Plan+ (Cashmore et al. 2016), Tarski (Francés, Ramirez, and Collaborators 2018), TFD (Röger, Eyerich, and Mattmüller 2008), and VAL (Howey and Long 2003).

3 Environments

Once installed, packages in PLANUTILS can be run by prefixing their invocation with `planutils run`. Alternatively, there are dedicated environments for a PLANUTILS shell, remote mirror, and local server.

3.1 PLANUTILS Shell Environment

For a more direct access to the PLANUTILS packages, we provide a special shell environment. Activating this environment exposes direct access to all PLANUTILS functionality, and invoking uninstalled packages dynamically loads them on demand:

```
$ planutils run lama d.pddl p.pddl
...
$ planutils activate

  Entering planutils environment...

(planutils) $ lama d.pddl p.pddl
```

3.2 Remote Mirror

In coordination with the newly-released Planning-as-a-Service project (AI Planning Community 2022), the functionality of PLANUTILS can be redirected to a central server. While this comes with reduced resource limits, this feature makes it very easy and safe for practitioners to run state-of-the-art planners on all operating systems:

```
$ planutils remote lama d.pddl p.pddl
```

3.3 Local Server

Finally, to interface with the online editor and VSCode plugin for PDDL (Muise 2016b; Dolejsi et al. 2018), the PLANUTILS functionality can be exposed via a localhost API. This allows practitioners to use resources of the host machine (and potentially custom planning software) with the leading PDDL editors.

```
$ planutils server
```

4 Virtualization

PLANUTILS is largely based on modern virtualization solutions. We describe the two key aspects of this here.

4.1 Singularity for Planners

Increasingly, modern IPC tracks are requiring planner submissions to be compiled in a Singularity image (Kurtzer, Sochat, and Bauer 2017). This allows organizers to run the planners without wrestling a web of conflicting dependencies between planners. PLANUTILS similarly recommends that self-contained systems be packaged as Singularity images for simplicity, and the machine where PLANUTILS is installed needs to be able to run these images. This is currently the case only for Linux.

4.2 Docker for Broad Access

To broaden the accessibility of PLANUTILS, we provide a pre-compiled Docker (Merkel 2014) image which allows quick and easy access to PLANUTILS for anyone with a Docker installation:

```
$ docker pull aiplanning/planutils
$ docker run -it --privileged \
    aiplanning/planutils
$ planutils --help
```

5 Getting Started

Getting started with PLANUTILS is as easy as installing the `planutils` Python package and running the `setup` command. Below we show the usage and output from a fresh installation running `lama`:

```
$ pip install planutils
$ planutils setup
$ planutils activate

  Entering planutils environment...

(planutils) $ lama d.pddl p.pddl

Package not installed!
  Download & install? [Y/n] Y
lama will be installed.

About to install the following
 packages: downward (36M), lama (20K)

  Proceed? [Y/n] Y

Installing downward...
INFO:    Downloading shub image
 45.84 MiB / 45.84 MiB
 [==========] 100.00% 7.30 MiB/s 6s

Finished installing downward (size: 46M)

Installing lama...
Finished installing lama (size: 20K)

Successfully installed lama!

Original command: lama d.pddl p.pddl
  Re-run command? [Y/n] Y

INFO     Running translator.
...
```

6 Summary

Even though PLANUTILS is still a very new project, it has already proved to be a powerful tool for rapidly prototyping planning solutions and offering direct access to the best tools our field has to offer. [1] The impact created by the initiative will be wide and long-lived, and we look forward to demonstrating PLANUTILS in its current form at ICAPS 2022.

[1] Used for hands-on part of the IJCAI 2021 and AAAI 2022 tutorials on AI Planning (https://github.com/IBM/grammar2pddl).

References

AI Planning Community. 2022. Planning-as-a-Service. https://github.com/AI-Planning/planning-as-a-service.

Alford, R.; Kuter, U.; and Nau, D. 2009. Translating HTNs to PDDL: A Small Amount of Domain Knowledge Can Go a Long Way. In Boutilier, C., ed., *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, 1629–1634. AAAI Press.

Alkhazraji, Y.; Frorath, M.; Grützner, M.; Helmert, M.; Liebetraut, T.; Mattmüller, R.; Ortlieb, M.; Seipp, J.; Springenberg, T.; Stahl, P.; and Wülfing, J. 2020. Pyperplan. https://doi.org/10.5281/zenodo.3700819.

Anaconda Inc. 2022. Anaconda Software Distribution. https://docs.anaconda.com/.

Benton, J.; Coles, A.; and Coles, A. 2012. Temporal Planning with Preferences and Time-Dependent Continuous Costs. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 2–10. AAAI Press.

Cashmore, M.; Fox, M.; Long, D.; and Magazzeni, D. 2016. A Compilation of the Full PDDL+ Language into SMT. In Coles, A.; Coles, A.; Edelkamp, S.; Magazzeni, D.; and Sanner, S., eds., *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2016)*, 79–87. AAAI Press.

Conitzer, V.; and Sha, F., eds. 2020. *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2020)*. AAAI Press.

Dolejsi, J.; Long, D.; Fox, M.; and Besançon, G. 2018. PDDL authoring and validation environment for building end-to-end planning solutions. In *Proc. of the 28th Int. Conference on Automated Planning and Scheduling (ICAPS)*.

Francés, G.; Ramirez, M.; and Collaborators. 2018. Tarski: An AI Planning Modeling Framework. https://github.com/aig-upf/tarski.

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

Howey, R.; and Long, D. 2003. VAL's Progress: The Automatic Validation Tool for PDDL2.1 used in the International Planning Competition. In Edelkamp, S.; and Hoffmann, J., eds., *Proceedings of the ICAPS 2003 Workshop on the Competition: Impact, Organisation, Evaluation, Benchmarks*.

Katz, M. 2018. Cerberus: Red-Black Heuristic for Planning Tasks with Conditional Effects Meets Novelty Heuristic and Enchanced Mutex Detection. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 47–51.

Katz, M.; and Sohrabi, S. 2020. Reshaping Diverse Planning. In (Conitzer and Sha 2020), 9892–9899.

Katz, M.; Sohrabi, S.; Samulowitz, H.; and Sievers, S. 2018a. Delfi: Online Planner Selection for Cost-Optimal Planning. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 57–64.

Katz, M.; Sohrabi, S.; and Udrea, O. 2020. Top-Quality Planning: Finding Practically Useful Sets of Best Plans. In (Conitzer and Sha 2020), 9900–9907.

Katz, M.; Sohrabi, S.; Udrea, O.; and Winterer, D. 2018b. A Novel Iterative Approach to Top-k Planning. In de Weerdt, M.; Koenig, S.; Röger, G.; and Spaan, M., eds., *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS 2018)*, 132–140. AAAI Press.

Kurtzer, G. M.; Sochat, V.; and Bauer, M. W. 2017. Singularity: Scientific containers for mobility of compute. *PloS one*, 12(5): e0177459.

Merkel, D. 2014. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux J.*, 2014(239).

Muise, C. 2016a. Planning.Domains. In *26th International Conference on Automated Planning and Scheduling, System Demonstrations and Exhibits*.

Muise, C. 2016b. Planning.Domains. In *The 26th International Conference on Automated Planning and Scheduling - Demonstrations*.

Python Software Foundation. 2022. Python Package Index - PyPI. https://pypi.org/.

Röger, G.; Eyerich, P.; and Mattmüller, R. 2008. TFD: A Numeric Temporal Extension to Fast Downward. IPC 2008 short papers, http://ipc.informatik.uni-freiburg.de/Planners.

Scala, E.; Haslum, P.; Magazzeni, D.; and Thiébaux, S. 2017. Landmarks for Numeric Planning Problems. In Sierra, C., ed., *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, 4384–4390. IJCAI.

Seipp, J.; Keller, T.; and Helmert, M. 2020. Saturated Cost Partitioning for Optimal Classical Planning. *Journal of Artificial Intelligence Research*, 67: 129–167.