# Planning Challenges 2.0

**Alexander Koller**  
Cluster of Excellence  
Saarland University  
Saarbrücken, Germany  
koller@mmci.uni-saarland.de

**Guido Tack**  
Programming Systems Lab  
Saarland University  
Saarbrücken, Germany  
tack@ps.uni-saarland.de

**Ronald P. A. Petrick**  
School of Informatics  
University of Edinburgh  
Edinburgh, Scotland, UK  
rpetrick@inf.ed.ac.uk

---

## Abstract

We revisit the structure of the International Planning Competition (IPC) and propose a new “rolling” competition as a permanent fixture for the planning community. A central competition server would continually evaluate planners and generate performance data, while members of the community could directly upload new planners and problem domains, with the server automatically reevaluating existing planners as necessary. Real-time results would continuously be available, while more formal competitions could be “run” at regular intervals by producing snapshots of current planners and problems. Researchers would have access to a central repository with the latest domains and performance data, running on a common hardware platform, plus a social networking forum for posting challenges, announcing new technologies, and discussion. Since much of the proposed infrastructure is not tied to a specific research area, this approach offers the possibility of joint ventures with other research communities.

## Introduction

The International Planning Competition (IPC)
[^1] recently celebrated its 10-year anniversary in 2008. During this time, six instances of the IPC have been run, each of which has helped drive the development of new planning technology by providing a regular forum for showcasing the latest trends in planner design, domain description languages, and challenging planning domains. Given the absence of a competition in 2010—breaking the competition’s established two year cycle for the first time since its inception—we feel the time is right to revisit the structure of the existing IPC, and suggest a new direction for the competition in the future.

This paper proposes a new direction for the IPC, building on the strength of previous competitions and inspired by competitions in other research communities. Most notably, we envision an interactive and user-centric competition, focused around a novel online infrastructure featuring:

- A “rolling” competition format with community-initiated track and problem specifications,
- Continuous evaluation of contributed planners on competition problem sets, with real-time access to performance data generated on a common hardware platform, and
- A centralised repository offering researchers access to the latest domains and planners, and a social networking forum for community news, discussion, and interaction.

Thus, while previous instances of the IPC have traditionally been event-based competitions, scheduled at regular intervals, we propose an ongoing competition as a permanent fixture for the planning community. In particular, members of the community would be able to upload new planners and problem domains to a central competition server that continually re-evaluates planners and generates performance data. Up-to-the-minute results would be centrally available, while more formal competitions could be “run” at regular intervals by producing snapshots of current planners and problems.

From a technical point of view, it is important to note that such an idea would not have been practical ten years ago during the time of the first IPC. It is only recently, due to advances in computing hardware, network bandwidth, and available software infrastructure, that a scenario such as this is now feasible. Even so, our proposed infrastructure nevertheless presents some technical challenges that must be addressed (e.g., security, accessibility, maintenance, etc.). However, we believe that for the most part, solutions to these problems can already be found by looking at similar (although smaller scale) competition efforts in the wider research community.

Most importantly, we believe this proposal offers significant benefits to the competition stakeholders: organisers, participants, and the wider planning community. Furthermore, since much of the infrastructure we describe is not tied to a specific research area, attempts to establish similar competitions could benefit from this approach, leading to joint ventures with research fields beyond planning—and the possibility of future interdisciplinary challenges.

## Levels of Community Involvement

We present our proposal for a new planning competition infrastructure in terms of an ascending scale of levels, each of which makes it easier or more rewarding for the community to participate in the competition process. As we will discuss below, we believe that the competition format we propose is applicable to fields beyond planning. We will therefore refer to systems, problem classes, and problem instances, instead of planners, planning domains, and planning problems.

Level 0: The current IPC

As background, and a starting point for our proposal, consider the process that underlies the current IPC. Every two years, the IPC committee looks for volunteers to organise and run the various tracks of that year’s competition. These organisers are responsible for assembling a number of problem classes and instances, by developing them themselves or soliciting them from the wider community. Sometimes this process results in, or coincides with, a new dialect of the Planning Domain Description Language, PDDL (McDermott et al. 1998), which describes for participants the supported language for that year’s competition. After the participants have developed their systems, they submit the source code to the organisers, who then compile and run the planners on the test machines. Performance data is collected and analysed. Finally, the organisers prepare and present a report, and the best-performing systems win awards.

Level 1: Problem collection

While this format for the IPC has been highly successful in the past ten years, there are also a number of ways in which it can be improved and streamlined. One such direction concerns the way in which problem classes are collected. There is no doubt that the existing collection of IPC domains is a fundamental driving force for the development of new planning algorithms. Unfortunately, the current selection process makes it difficult for users of these systems, who may come from different communities, to get a new application-motivated domain accepted as an IPC benchmark. This is unsurprising, given that IPC organisers must be able to justify their selection of domains, and picking untested domains from other communities is not the best way to do this. However, this leads to a selection process that is less dynamic than it could be, and more restrictive for users wanting to contribute suggestions. Conversely, to keep the number of domains that must be tested in each round of the competition manageable, older domains are sometimes “retired”, i.e., removed from subsequent competitions. This comes with a risk of a false sense of progress, in which retired domains are remembered as “solved” although they were removed for completely independent reasons. Finally, it has been argued (Hoffmann 2007) that IPC organisers may be tempted to favour the development of new types of planning problems and new dialects of PDDL over trying to find challenging benchmark problems for existing versions of PDDL. Such approaches run the risk of “reinventing” past problem classes (albeit in different guises), or redirecting investigation away from current challenges.

One way to alleviate these problems is by introducing a web repository where users can upload problem classes. This revisits Hoffmann’s (2007) idea to create a community-driven repository for “tough nuts”—i.e., instances that were shown to be hard to solve in the past—and takes it one step further: in our proposal, all IPC problem classes are provided by the community, much in line with the proposal by Roberts and Howe (2007). Such a repository offers a principled, publicly documented and accessible avenue for proposing new problem classes, which benefits both application-oriented users of planning technology and researchers who invent tough challenges for existing PDDL dialects. It also makes it easy to record retired problem classes alongside active ones, and keep them all accessible.

Problem classes and instances can be submitted by any registered user. Submitting a problem class should involve three pieces of data. First, the user should submit a collection of instances of the problem class or a generator for such instances. Second, the user should specify some metadata including the class name and a human-readable description, possibly with links to the applicable literature, complexity results, and so on. Third, the user should specify the type of problem class, which determines what systems the data is suitable for, i.e., the language in which the problem is specified (such as a particular dialect of PDDL) and the type of planning problem (e.g., propositional, temporal, uncertainty). If it is possible to convert between one specification language and another automatically, this conversion could be performed each time a problem is submitted. In order to avoid abuse, the number of instances should be limited (e.g., per hour), and a simple validity check should be possible (e.g., running a parser to verify that the format is PDDL).

Level 2: Continuous evaluation

Given that all the domains are available on a server, why stop there? We propose to make it possible to upload the planners to the same server as well, and then continuously and automatically run the planners on the domains. While the ability to upload domains in Level 1 represents a conservative extension of the existing IPC infrastructure, it is here that we depart more radically from the existing format: instead of a regular competition with organisers who run the planner evaluation every other year, we propose a rolling evaluation scheme, in which the current evaluation results would be available online at any given time.

Of course, several issues need to be addressed to make this possible. The first is how we can allow users to upload their systems and still expect to run and maybe compile them automatically, without requiring the manual intervention of an organiser if things break. This problem can be solved by imposing strict requirements on the execution environment of the planner. One way in which we could envision this is by defining a single hardware and software environment in which systems will be evaluated, and making it available for download as a virtual machine image. Developers can then run this image on their own machine, using a virtual machine such as VirtualBox, and ensure that their system works in the exact environment in which it will be evaluated. This would even make it technically possible for participants to submit not the source code, but a binary executable (perhaps along with support files). However, even if we require the submission of source code, the move to a precisely defined environment that contains only standard versions of tools and libraries and on which the code must compile without human intervention enforces at least a certain degree of portability from which users will benefit when they download the systems. Even when the environment

---

2http://www.virtualbox.org/
changes to reflect updates in the standard development environments (e.g., a new version of the C compiler), old images could remain available, and so old systems would continue running until the virtual machine emulator itself becomes unavailable or incompatible.

We propose that every newly submitted system be run automatically on all available problem instances, and every newly submitted problem instance be sent to all available systems. The scheduling of this process can be under the control of an automated system for continuous integration, such as Hudson. The evaluation infrastructure would send the problem data and the system to the test machine, run the system, measure performance statistics (including which problems they could solve and how much time this took them; see below), and store the results in a database. Users would be able to view the results in dynamically generated reports on the main website; there may also be functionality to customise these reports to focus on certain types of problems. Because all systems are run on all problem instances, and users can ensure that challenging problem instances are included in the competition, the chance of systems being evaluated on only a small and somewhat arbitrary selection of problem instances (Gregory and Lindsay 2007) can be avoided.

In order to minimise security risks, which can arise when user-submitted programs are run automatically, the test machines need to be sandboxed quite tightly. In particular, the machines do not need to be able to initiate any network connections, beyond being able to enter performance measurements into a database, and can work on virtual file systems that are deleted after each execution. By further imposing resource limits on the number of available processors, memory, and processor time, and disk space, we believe the potential for abuse is limited. It can be limited further by only allowing system uploads to registered users.

While we see our proposed system’s ability to generate up-to-date reports automatically as a major strength, there may still be a benefit in maintaining the current style of regular competitions at discrete timepoints, e.g., to encourage the development of faster systems through awards. Such a competition could easily be based on our system, by announcing a deadline and a specified subset of problem classes. The selection of this subset could be informed, among other things, by the report on problems that many systems had difficulty solving. Participants could then upload their systems at their leisure; “running” the competition would amount to taking a snapshot of the performance reports on the selected problems when the deadline comes and announcing the results. By recording timestamps whenever a performance measurement is made, this snapshot could remain available on the website simply as one special report for a certain point in time. In such a case, the authors of systems should have the option of marking them as “non-competing” to exclude them from the competition snapshot, and maybe also from the up-to-date performance reports.

Level 3: Social networking

As a final step, we justify the “2.0” in the title of our paper by proposing the addition of mechanisms for social networking to the continuous evaluation website.

The simplest such mechanism is to add Wiki pages to the evaluation website. This would allow the users who upload problem classes to document these problems and link to existing literature, it would allow the developers of systems to document these systems, and it would make it easy for the organisers of regular competitions to add more information to the snapshot reports. The exchange of ideas and opinions can be further facilitated by providing a discussion board which links into these Wiki pages.

A second social networking tool which we believe would be useful in the context of a planning evaluation website is to let users tag problem classes and vote on them. Any registered user can be allowed to comment on any problem class and vote for how relevant, important, or exciting they find it—say, using a one to five star system as on Amazon. This has a number of crucial advantages. First, an incoming user or a competition organiser can take the average score as an indication for the problems that are considered important in the community; imagine a “top ten” list of problem classes and systems on the website’s front page. Next to the difficulty of problems, this can be another factor in the selection of problem classes for a competition snapshot. Second, a relevance score for problem classes can also serve as a guideline for the scheduling of system runs: assuming that there is not enough computing power to run all incoming planners on all problem instances in real time (e.g., just before a competition deadline), the runs on popular domains can be prioritised.

Evaluation Measures

The continuous evaluation infrastructure we have presented is fundamentally agnostic to the type of concrete evaluation measures that it collects. In particular, traditional measures such as the percentage of problems solved per system, average solution quality, average runtime and memory use, and search tree sizes can be collected—with the caveat that the test machines must be configured so as to allow comparable runtime measurements. Furthermore, popularity scores from the social networking aspect can be incorporated, or different measures could be combined into an overall score.

Implementation

The obvious criticism that can be brought against this proposal is that of cost—both in terms of implementing the necessary software and in terms of running the hardware on which the continuous evaluation is performed.

We believe that the actual hardware cost will be bearable. While the idea of running all systems on all problem instances sounds intimidating at first, we don’t anticipate that there will ever be more than a couple dozen current versions of systems entered in the evaluation process. This means that any new domain that is entered will not be attempted to be solved too many times. Conversely, while there may

---

3http://hudson-ci.org/
(hopefully!) be a significant number of domains in the system, and every new system must try to solve all problem instances whose types it claims to be able to process, this cost can also be controlled by prioritising popular or hard domains. A single computer should be sufficient for these needs for the most part; in times of high demand (e.g., just before a competition deadline), the work can be automatically distributed over “slave” test machines by an off-the-shelf continuous integration engine.

On the other hand, we expect that implementing the software for running the continuous evaluation and presenting the results on a website would not be a prohibitive effort either. Most of the components of the system (continuous integration server, virtual machines, Wiki) already exist as open-source off-the-shelf components. We estimate that a prototype implementation of a complete system could be built in a person-month of work; surely this is the amount of work for which funding or volunteers could be found.

Applicability and Limitations

One advantage of the evaluation infrastructure proposed above is that its applicability is not limited to the evaluation of planners. Many reasoning tasks, including constraint solving and theorem proving, are evaluated by running solvers over problem instances and reporting performance measures. At the level of abstraction at which our continuous evaluation system could be implemented, there is no fundamental difference between the problem type “deterministic planning problem in PDDL 1.0” and the problem type “theorem of first-order logic in TPTP format”—in both cases, the system can collect problem classes and run them through solvers that declare themselves to accept such inputs. In fact, other fields are already ahead of the planning community in maintaining problem repositories as on Level 1 above (Sutcliffe 2009; Gent and Walsh 1999; Asuncion and Newman 2007). This means that there is a potential for sharing development and maintenance costs for the evaluation platform among different communities.

However, there are of course limitations to the type of problem for which it makes sense to run a community-driven continuous evaluation. For one thing, it must be acceptable in the community that every system runs on the same hardware and software configuration, as implemented in the virtual machine images. On the other hand, our system makes most sense where the ratio of problem classes to systems is relatively high. Neither of the two is true, for instance, for most natural language processing competitions: different research groups tend to use a variety of programming languages and development environments to build their systems, and there are typically many more systems in any competition than there are problem classes (e.g., text corpora). However, our approach is completely consistent with the requirements of a community where solving a single hard problem instance for the first time counts as an achievement, such as in theorem proving.

Conclusion

The new direction we propose for the IPC has significant advantages for each of the competition’s three main groups of stakeholders: the organisers, the participants, and the audience. Organisers benefit from a mostly automated platform that is maintained by the community, which should greatly reduce the effort needed to run a competition. Participants benefit from a well-defined environment for running their systems, short feedback times for system evaluation, and a standardised test and benchmark platform for experimenting with novel algorithms. The audience—in this case the wider planning and scientific communities—benefits from improved discussion and reporting facilities. Since the scientific value of competitions is to find out why a system performs the way it does, not only that it performs a certain way, enabling users to contribute to the competition process reduces the chance that essential avenues of investigation and evaluation are missed. Instead, a robust and active competition community only serves to enhance the wider planning community as a whole.

References


