29th International Conference on
Automated Planning and Scheduling

July 10–15, 2019, Berkeley, CA, USA

WIPC 2019

Proceedings of the

2019 Workshop on the International Planning Competition

Edited by:

Álvaro Torralba, Florian Pommerening, Thomas Keller, Amanda Coles, and Andrew Coles

Organization

Álvaro Torralba
Saarland University, Germany

Florian Pommerening
University of Basel, Switzerland

Thomas Keller
University of Basel, Switzerland

Amanda Coles
King's College London, United Kingdom

Andrew Coles
King's College London, United Kingdom

Contents

Hierarchical Planning in the IPC

G. Behnke*, D. Höller*, P. Bercher*, S. Biundo*, D. Pellier†, H. Fiorino†, and **R. Alford‡**
*Institute of Artificial Intelligence, Ulm University, 89081 Ulm, Germany
{gregor.behnke, daniel.hoeller, pascal.bercher, susanne.biundo}@uni-ulm.de
†University Grenoble Alpes, LIG, F-38000 Grenoble, France
{damien.pellier, humbert.fiorino}@imag.fr
‡The MITRE Corporation, McLean, Virginia, USA
ralford@mitre.org

Abstract

Over the last years, the amount of research in hierarchical planning has increased, leading to significant improvements in the performance of planners. However, the research is diverging and planners are somewhat hard to compare against each other. This is mostly caused by the fact that there is no standard set of benchmark domains, nor even a common description language for hierarchical planning problems. As a consequence, the available planners support a widely varying set of features and (almost) none of them can solve (or even parse) any problem developed for another planner. With this paper, we propose to create a new track for the IPC in which hierarchical planners will compete. This competition will result in a standardised description language, broader support for core features of that language among planners, a set of benchmark problems, a means to fairly and objectively compare HTN planners, and for new challenges for planners.

Introduction

When the International Planning Competition (IPC) started out in 1998 it aimed to include both classical and hierarchical planners as competitors (McDermott 2000). To provide a fair competition and foster further research in planning, a unified description language for planning problems, the Planning Domain Definition Language (PDDL), was created. The main objective of PDDL was to separate the description of the physics of a domain from additional advice. Planners should operate on the physics of the domains only. This way the competition shows how good the underlying domain-independent techniques are for solving planning problems based on the description of a domain's physics. Notably, PDDL also included features for expressing hierarchical planning problems. Hierarchical planning was not included in the first IPC, as there were no competitors.

The second and third IPC subsequently included a separate track for "Hand-Tailored Systems", where the planner was allowed to have additional advice for each individual domain. This was within the spirit of hierarchical planning at the time (see e.g. SHOP's participation (Nau et al. 1999)). Hierarchical structures in a planning problem were – at the time – almost exclusively seen as additional advice given to the planner in order to speed up planning. The hierarchy was viewed as specifying pre-defined recipes for a given objective. After the third IPC, no further attempt was made to include hierarchical planning into the IPC.

Since the third IPC in 2002, the research in hierarchical planning has progressed significantly. Especially the view what hierarchical planning is, has changed sharply. A hierarchy over a planning problem is not (only) a means to provide advice to the planner, but a general means to specify parts of the domain. Just like non-hierarchical structures, it may be used to model advice (as done before), but also to describe additional physics of the domain. Research in hierarchical planning has, over the last decade, mostly focussed on a single formalism: Hierarchical Task Network (HTN) Planning (Erol, Hendler, and Nau 1996)[1]. The physics expressible in HTN planning cannot be equivalently expressed in classical planning (Höller et al. 2014; Höller et al. 2016). In fact, HTN planning allows for expressing undecidable problems, like Post's Correspondence Problem or Context-free Grammar Intersection (Erol, Hendler, and Nau 1996). Based on the researcher's focus on a single, theoretically understood formalism, we think the time is right for an IPC track in which HTN planners compete. Creating such a competition was one of the motivations behind organising the Workshops on Hierarchical Planning at ICAPS 2018 and 2019. A discussion on the details of a (potential) IPC track for HTN planning is planned for this year's workshop.

The HTN planning community lacks a strong unifying force like the IPC is for classical planning. Due to the lack of a competition in HTN planning, there is no precisely agreed-upon semantics for modelled planning problems[2], no unified input language readable by all planners, and no standardised benchmark to compare planners. A competition should resolve these problems and provide a solid basis for future work, as was the case for classical planning. We therefore propose to add a new track to the IPC in which HTN planners will compete. In this paper, we first give a brief overview of HTN planning, report on results of an online questionnaire on an HTN IPC track, and then present the details of our prosed HTN IPC track.

[1]Note that there is also research within hierarchical planning that does not qualify as HTN planning, such as work on HGN and GTN planning (Shivashankar et al. 2013; Alford et al. 2016b). See Bercher, Alford, and Höller (2019) for an overview.

[2]Although there is a theoretical formalism almost all planners add their individual bells and whistles making them incomparable.

HTN Planning – State of the Art

We start by giving a quick overview over HTN planning, by reporting on theoretical and practical results related to it.

Theory

While classical planning has only one type of tasks – actions – HTN planning distinguishes two types: primitive tasks (also called actions) and abstract tasks. Only actions hold preconditions and effects, which are defined in the usual (STRIPS or ADL) way. In contrast to actions, abstract tasks cannot be executed directly. Instead, each abstract task A has a set of associated decomposition methods (A, tn) that allow for achieving A by performing the tasks (that may, again, be primitive or abstract) contained in tn. Here tn is a task network, which gives HTN planning its name. A task network is a partially ordered multi-set of tasks.

The objective in HTN planning is given by an initial state and an initial abstract task (replacing the state-based goal definition in classical planning[3]). A solution is any sequence of actions that is executable in the initial state, provided it can be obtained from the initial abstract task by repeatedly applying decomposition methods. In this structure, HTN planning is very similar to formal languages, where terminals correspond to actions and non-terminals to abstract tasks (Höller et al. 2014; Barták and Maillard 2017).

A simplistic formal description of HTN planning was proposed by Geier and Bercher (2011). An equivalent formulation in terms of combinatorial grammars was given by Barták, Maillard, and Cardoso (2018). The simplistic formalism by Geier and Bercher is theoretically well understood, witnesses by the multitude of papers based on it.

The plan existence problem in HTN planning in its general form is undecidable (Erol, Hendler, and Nau 1996). There are, however, several restrictions to the formalism that make it decidable. The most notable one is totally-ordered HTN planning. Here, all task networks in decomposition methods must be sequences of tasks instead of partially-ordered sets. In contrast to general HTN planning, ground, totally-ordered HTN planning is EXPTIME-complete (Erol, Hendler, and Nau 1996).

Planners

In the past, a multitude of HTN planners have been developed. This includes the older systems which were mostly based on uninformed search, like SHOP (Nau et al. 1999), SHOP2 (Nau et al. 2003), or UMCP (Erol, Hendler, and Nau 1994). Recently, the portfolio of available methods to solve HTN planning problems has increased significantly. Many ideas developed for classical planning have been transferred to HTN planning and proved successful there. This lead to a wide range of (partially) available HTN planners.

- FAPE (Dvorak et al. 2014), a temporal HTN planner with strong pruning techniques
- PANDA (Bercher et al. 2017), a plan-space planner using heuristic search
- PANDApro (Höller et al. 2018; Höller et al. 2019b), a progression-based planning system using heuristic search
- GTOHP (Ramoul et al. 2017), a planner based on intelligent grounding and blind search
- HTN2ASP (Dix, Kuter, and Nau 2003), a planner that translates (totally-ordered) HTN planning problems into answer set programming
- HTN2STRIPS (Alford et al. 2016a), a planner translating HTN planning problems into a sequence of classical planning problems
- totSAT (Behnke, Höller, and Biundo 2018) and Tree-Rex (Schreiber et al. 2019), planners that translates (totally-ordered) HTN planning problems into propositional logic
- partSAT (Behnke, Höller, and Biundo 2019a; 2019b), a planner based on a translation into propositional logic

It is noteworthy that four of the planners (GTOHP, totSAT, Tree-Rex, and HTN2ASP) are specifically designed for solving totally-ordered HTN planning problems. Further, HTN2STRIPS is based on an older encoding of totally-ordered HTN planning problems into classical planning (Alford, Kuter, and Nau 2009).

Online Questionnaire

In order to better understand the needs and interest of the community when organising a deterministic HTN IPC track in 2020, we conducted a short online questionnaire and sent the link for participation to the planning community.

We obtained 23 responses from 20 different research groups. 80% of the respondents would like to participate in an HTN IPC track in 2020. Those not wishing to participate often justified their refusal with the lack of HTN standards (language, benchmarks, tools). We would argue that starting the competition is the only way to produce such a standard in the first place. As such, organising an HTN IPC competition seems to respond to the needs of the community.

The main motivation for those wishing to participate is to be able to objectively compare the performance of HTN planners. However, this was not the only reason given by respondents. People also wished to increase the expressiveness of the PDDL language by adding HTN language standards.

We also asked what the respondents thought to be absolute necessities for the competition. The four top answers ranked in order of importance are:

1. a standard language definition for HTN planning,
2. an accessible HTN benchmarks repository,
3. an HTN plan validator, and
4. an HTN parser.

In terms of tracks, most people think that optimal and satisfying should be enough for a first IPC competition. Some people also expressed interest in participating in a temporal track. For the first HTN IPC, we propose to restrict the competition to a non-temporal setting, with the option of a temporal track to be added in the future. We think that the competition should focus on the core of HTN planning and thus should compare the planners on the absolute essentials.

In terms of metrics that should be used to compare HTN planners, opinions are divided. 50% of the respondents think

[3]Note that every classical planning problem can be converted into an equivalent HTN planning problem (Erol, Hendler, and Nau 1996).

that the IPC score is suitable for an HTN IPC track. 50% think that we need to devise new metrics, which they could propose as free text answers. Respondents proposed to use (1) the number of backtracking operations performed by the planner, (2) the depth the explored search space, (3) the size of the method library used by the planner, and (4) the expressivity and explicability of the solution.

We propose to keep the IPC score as opposed to these proposals. First, we argue that all planners in the competition *must* use the same HTN method library to ensure that the results of the competition are correct, objective, and fair. If every planner would be allowed its own set of methods, we would not measure the performance of the planner, but the ingenuity of the modeller. Furthermore, the set of solutions the problems encodes would differ from planner to planner – which makes little to no sense, as a comparison between the mechanics of planners is not possible anymore.

Using either the number of backtracking operations or the depth of the explored search space would restrict the competition to search-based planners, as e.g. SAT-based planners can't produce these measures. We think that these two metrics stem from the idea that the planners may use their own HTNs for the domains, as both the number of backtracking operations as well as the depth of the search space are metrics to measure the "difficulty" of an HTN domain. Further, to measure these metrics, we would have to rely on the information reported by the planners about their internal search, which is somewhat strange for an objective competition. Similarly, measuring the explicability of solutions objectively is hard or even impossible at the moment.

In terms of HTN planning language, a majority of people thinks it is better to add hierarchical planning concepts to the PDDL language rather than redefining a new language from scratch. We elaborate on this in the next section.

Concerning benchmarks, most respondents wish to participate to the definition of HTN benchmarks. We asked respondents which types of domains they wished to see as part of the benchmark set (see Fig. 1). Note that these answers have to be considered with care, as most of the mentioned types of domains are also domains used in the classical IPC. HTN planning is *not* a means to solve classical planning problems faster, but a complex combinatorial problem in its own right. There are structural restrictions (e.g. PCP and Grammar Intersection domains) to plans that cannot be expressed by preconditions and effects of actions, while it is possible to formulate them in an HTN domain (Erol, Hendler, and Nau 1996; Geier and Bercher 2011; Höller et al. 2014). Further, the domain's hierarchy can express physics that is not modelled in the primitive actions and is thus an integral part of the problem definition. For example, in a transport domain, the location of each truck can be fully modelled in an HTN problem without introducing state variables pertaining to the location of the truck. Without the hierarchy, the primitive action theory in this model makes no sense at all. As such, the mentioned domain proposals should be viewed as potential topics for modelling domains with highly complex restrictions on plans and not as the suggestion that the domain should express the same mechanics as its classical counterpart.

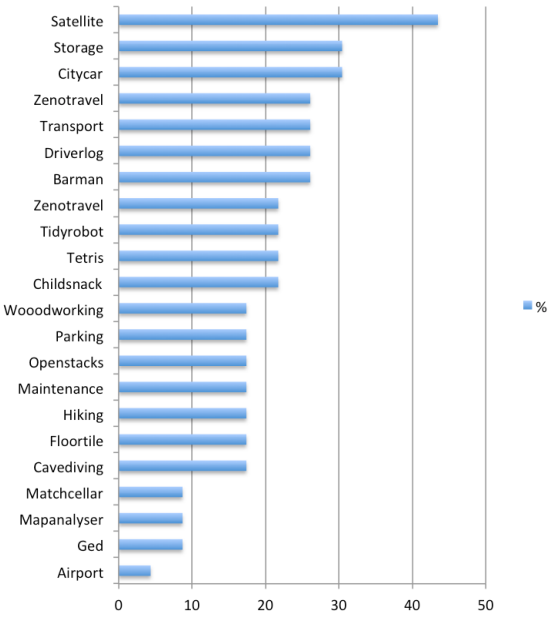


Figure 1: Types of domains proposed by respondents for HTN IPC track

Common Description Language

Organising the first IPC track on HTN planning also involves defining a fixed input language for all participating planners. For classical planning, there is one universally accepted description language – PDDL. The situation for HTN planning is drastically different. Of the planners mentioned in the previous section, PANDA, PANDApro, totSAT, and partSAT accept the same input language. Similarly GTOHP and Tree-Rex have a common input language, which is distinct from PANDA's. All other planners use their own individual description languages. Both groups of planners using the same input language also share a common parser and grounder. The two formats are somewhat incompatible and the format of GTOHP and Tree-Rex does only support totally-ordered HTN planning problems.

Some of the description languages are based on PDDL, while others, like SHOP's language and ANML (Smith, Frank, and Cushing 2008) are drastically different. We think that a language based on PDDL is the most sensible way to define the HTN description language, as it will allow for close contact between the HTN and classical planning communities (tools for preprocessing like Fast Downward's grounder which transforms the planning problem into the SAS+ format (Helmert 2009) might, e.g. be used in both communities). A proposal for an HTN description language (Höller et al. 2019a) has been accepted at this year's Workshop on Hierarchical Planning at ICAPS.

The proposed language is based on the STRIPS part (language level 1) of the PDDL 2.1 language definition (Fox and Long 2003). In this paper, we want to introduce the extensions made and explain some of the design decisions. In the original paper (Höller et al. 2019a), we give a much broader discussion, especially with respect to several alternative def-

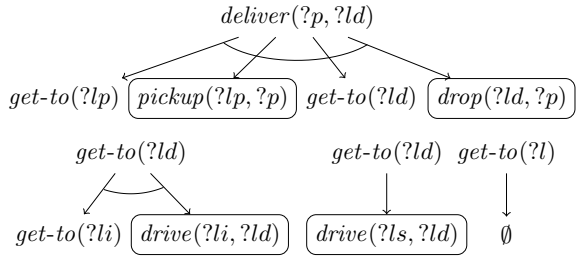


Figure 2: The method set of a simple transport domain. Actions are given as boxed nodes, abstract tasks are unboxed. All methods are totally ordered (source of figure: Höller et al., 2019a).

initions from related work and also provide a full EBNF definition of the entire language.

We want to introduce the language using a simplistic transport domain that is illustrated in Figure 2. There is only a single transporter that has to deliver packages. The method set is designed to exemplify the features of the language (so we know that there are more compact equivalent definitions). The overall deliver task can be decomposed (by the method given at the top) into a *get-to* task that moves the transporter to the package, a *pick-up* action, another *get-to* task that makes the transporter move to the position the package shall be delivered, and a *drop* action. For the *get-to* task, there are three methods (given at the bottom, from left to right): the first one is recursive and makes the transporter get to an intermediate position from which the target position can be reached directly by a primitive *drive* action, the second one can be used when it is already directly reachable by a *drive* action, and the third one can be used when the transporter is already at its destination, it decomposes the *get-to* into an empty task network. The search is started with one or more *deliver* tasks.

When we look at the proposed input language, the beginning of the domain definition is equal to PDDL, defining the domain name, a type hierarchy, and predicates.

```
1 (define (domain transport)
2   (:types location package - object)
3   (:predicates
4      (road ?l1 ?l2 - location)
5      ...)
```

The first element we had to add is the definition of *tasks*. There are two common ways to define them: as an explicit enumeration, or, by simply *using* the tasks in the method specifications and defining the set of abstract tasks as the union of all tasks used in methods. We decided to require an explicit definition of abstract tasks and tried to keep it close to the one of actions in PDDL (in fact, both actions and abstract tasks are quite similar in HTN planning) due to the following reasons:

- In our opinion, an implicit definition is against the design spirit of PDDL. Take the predicate definitions as an example. It would be sufficient to omit the explicit definition of predicates and just use them in action definitions. PDDL opted for the first variant, an explicit definition.
- There are hierarchical planning approaches where abstract tasks hold preconditions and effects. A definition as given here allows for a simple extension to support such approaches.
- Lastly, there are benefits to being able to explicitly model the types of parameters of abstract tasks (see how tasks are used in method definitions).

The following listing defines the two abstract tasks used in the transport example.

```
6   (:task deliver :parameters (?p - package
        ?l - location))
7   (:task get-to :parameters (?l - location))
```

There is a single method decomposing the *deliver* task:

```
8   (:method m-deliver
9     :parameters (?p - package
          ?lp ?ld - location)
10    :task (deliver ?p ?ld)
11    :ordered-subtasks (and
12      (get-to ?lp)
13      (pick-up ?ld ?p)
14      (get-to ?ld)
15      (drop ?ld ?p)))
```

The method definition starts with its name (here: `m-deliver`) and is followed by the parameter definition.

The parameters could, again, be either defined explicitly (like we did), or implicitly by just using them. Having an explicit definition has two advantages:

- Having an explicit definition allows for consistency checks by the planning system.
- By having the method parameters specified explicitly, one can restrict the applicability of a method via the type definition. As an example, consider a transport domain with several types of packages. Some of them might need special ways to be delivered, e.g. hazardous materials.

In the proposed language, all variables used in a method must be declared as parameters of that method – similar to the parameters of PDDL actions. This includes variables used for the specification of the abstract task that is decomposed, in the specification of the subtasks, and the constraint set.

Next, the task that the method decomposes is specified (line 10). It has to be defined in the domain as given before.

The method definition closes with the subtasks (starting in line 11). In the given method, the tasks shall be totally ordered. To enable a compact domain definition, the modeller can use the `:ordered-subtasks` keyword to indicate that all subtasks shall be totally ordered in the order in which they are specified. As subtasks, any action or task defined in the domain can be used, while their arguments can be constants and any of the parameters of that method.

In the next method, we see how to define the order of subtasks explicitly.

```
16  (:method m-drive-to-via
17    :parameters (?li ?ld - location)
18    :task (get-to  ?ld)
19    :subtasks (and
```

```
20        (t1 (get-to ?li))
21        (t2 (drive ?li ?ld)))
22      :ordering (and
23        (t1 < t2)))
```

The subtasks are "marked" with IDs (here: `t1` and `t2`). The order is specified using these IDs. Supporting these two ways to specify ordering, we enable a compact definition of totally ordered networks (using the first variant) without loosing the expressivity to specify arbitrary partial order.

Many HTN planners allow for method preconditions (see Höller et al. (2019a) for a discussion of this feature). We decided to include them as well.

```
24    (:method m-already-there
25      :parameters (?l - location)
26      :task (get-to ?l)
27      :precondition (tAt ?l)
28      :subtasks ())
```

Here, the *get-to* task may be achieved by doing nothing when the transporter is already at its final position. This is checked by the state-based precondition (line 27).

HTN planners from the literature support various forms of conditions, like e.g. equality and inequality of variables, constraints on the types, but also conditions on the state that need to hold between two tasks. Though we decided to keep the initial formalism simple, we added a `constraints` section to our definition that can be seen in the following listing (line 32).

```
29    (:method m-direct
30      :parameters (?ls ?ld - location)
31      :task (get-to ?ld)
32      :constraints
33        (not (= ?li ?ld))
34      :subtasks (drive ?ls ?ld))
```

In the first language version, only equality and inequality are supported, but the presence of the constraints section allows for an easy extension in future language versions.

The action definitions remain as defined in PDDL2.1.

```
35    (:action drive
36      :parameters (?l1 ?l2 - location)
37      :precondition (and
38        (tAt ?l1)
39        (road ?l1 ?l2))
40      :effect (and
41        (not (tAt ?l1))
42        (tAt ?l2)))
43    ...)
```

Another important change is made to the problem definition. Here, the initial task network is specified (starting in line 6), its definition is similar to the definition of method sub-networks (therefore we included the empty `:ordering` and `:constraints` in the next listing to show the similarity, these might, of course, be omitted).

```
1  (define (problem p)
2   (:domain transport)
3   (:objects
4    city-loc-0 city-loc-1 city-loc-2 -
         location
5    package-0 package-1 - package)
6   (:htn
7    :tasks (and
8     (deliver package-0 city-loc-0)
9     (deliver package-1 city-loc-2))
10   :ordering ()
11   :constraints ())
12  (:init
13   (road city-loc-0 city-loc-1)
14   (road city-loc-1 city-loc-0)
15   (road city-loc-1 city-loc-2)
16   (road city-loc-2 city-loc-1)
17   (at package-0 city-loc-1)
18   (at package-1 city-loc-1)))
```

Another point we want to highlight is the specification of the problem class given in line 6. It is specified by the keyword `:htn`. There are many slightly different problem classes in hierarchical planning (see Bercher, Alford, and Höller (2019) for an overview). Some allow, e.g., for task insertion. The explicit definition in the problem file allows to extend the language standard to other classes than common HTN planning.

Tracks

As in the IPC for deterministic, classical planning, we propose to run competitions for optimal, satisficing, and agile planning. Further we propose to split the competition between the types of problems handled by the planners. Since there is a large group of existing HTN planners that is restricted to totally-ordered HTN planning problems, we propose to include a separate track for those planners in which all input problems are totally-ordered. Naturally, all HTN planners that are capable of handling general HTN planning problems can take part in a totally-ordered competition, but should be at a disadvantage. In a second set of sub-tracks, the planners that are able to solve general, i.e. partially-ordered, HTN planning problems will compete. As a result, we propose six tracks: optimal totally-ordered, satisficing totally-ordered, agile totally-ordered, optimal general, satisficing general, and agile general.

Timeline and Organisation

We propose to roughly follow the usual schedule of the IPC.

May – July 2019	Agreeing on a common input language for all planners.
July 2019	Announcement of the track Call for domains Call for expression of interest
October 2019	Registration deadline
November 2019	Demo problems provided
January 2020	Submission of preliminary planner versions
February 2020	Domain submission deadline
April 2020	Final planner submission deadline
May 2020	Paper submission deadline
May 2020	Contest run
June 2020	Presentation of the results at ICAPS 2020

As did the first IPC track on unsolvability (Muise and Lipovetzky 2015), we also expect that there will be many technical issues with the submitted planners. We assume that most of them will be centred around correct support of the input language, for which issues will usually take some time to debug and fix. Further, HTN planners have to provide the decompositions they used as their output as well – in order to be able to verify their solutions in time. Without these decomposition, verification is $\mathbb{NP}$-complete (Behnke, Höller, and Biundo 2015) and thus may take a long time. We will use the preliminary submissions of the planners to validate that their outputs are correctly formatted and can be verified against the domain. As such, we propose a first submission of the planners early on, so that we can test them sufficiently before the actual competition.

Scoring and Setting

For scoring the planner, we will adopt the metrics used in the last deterministic, classical IPC. We further propose to use the same technical setting (1 core, 8 GB of RAM, and 30 minutes). All planners will be provided with the same HTN planning domain, i.e. set of primitive actions, abstract tasks, and decomposition methods, the same planning problem, i.e. initial state and initial abstract task, and are not allowed to have any additional domain-dependent information. They have to output both the solution, i.e. a sequence of primitive actions, as well as a witness that this solution is obtainable via decomposition form the initial abstract task.

Domains

As noted before, there is no large set of available benchmarking domains for HTN planners, which would have to be created for the competition. The IPC has so far relied for the domains in the competition on an open call for the community to submit domains. We propose will also call for such community-provided domains (once a input language has been fixed). In addition, we propose to add a new mode of providing benchmark domains that has been recently adopted by the SAT community: Bring Your Own Benchmark (BYOB). The SAT Competition 2017[4] and 2018[5] used BYOB. In it, each competing program – in our case planner – is required to submit a domain with 20 instances. Of these 20 instances, the planner of the submitter must be able to solve at *most* 10. This requirement forces submissions of domains that are not advantageous to the planner of the submitter. This encourages the submission of problems that are of medium difficulty for the individual planner. We argue that they will both provide a good basis for comparison against other planners as well as a good starting point for future scientific investigations based on the difficulties in them.

From a content perspective, we especially encourage submissions of domains that pose problems which cannot be expressed in classical planning, i.e. in PDDL[6]. Such domains exists, like PCP and Grammar Intersection, but it would be interesting to see where HTN planning can further use its high expressive power.

[4]https://baldur.iti.kit.edu/sat-competition-2017/

[5]http://sat2018.forsyte.tuwien.ac.at/

[6]without numbers

Discussion

With a new IPC track on HTN planning, we think that the research in hierarchical planning will be more focussed and successful in the future. As a result of the competition, we expect that

- the HTN community will agree upon a set of core features supported by every planner and an input language for it, which is important for users of HTN planning, as well as for comparing systems against each other.
- a set of benchmark domains will be available, allowing for better judging and fairer comparisons of planners than is currently possible.
- a core of (relatively) error-free software to be used by many planners will emerge, like Fast Downward (Helmert 2006) for classical planning, allowing for both easy use by planning researchers and users of planning technology.
- hints for future research in HTN planning will be given.

While the first three points are valuable to the community and outsiders wanting to use HTN planning, we want to emphasise the fourth. The results of planning competitions will regularly show the weaknesses of the so-far developed approaches and techniques. These weaknesses are valuable information and point out where planners can and should be improved in the future.

References

Alford, R.; Behnke, G.; Höller, D.; Bercher, P.; Biundo, S.; and Aha, D. W. 2016a. Bound to plan: Exploiting classical heuristics via automatic translations of tail-recursive HTN problems. In *Proc. of the 26th Int. Conf. on Autom. Plan. and Sched., (ICAPS 2016)*, 20–28. AAAI Press.

Alford, R.; Shivashankar, V.; Roberts, M.; Frank, J.; and Aha, D. W. 2016b. Hierarchical planning: relating task and goal decomposition with task sharing. In *Proc. of the 25th Int. Joint Conf. on AI (IJCAI 2016)*. AAAI Press.

Alford, R.; Kuter, U.; and Nau, D. 2009. Translating HTNs to PDDL: A small amount of domain knowledge can go a long way. In *Proc. of the 21st Int. Joint Conf. on AI (IJCAI 2009)*, 1629–1634. AAAI Press.

Barták, R., and Maillard, A. 2017. Attribute grammars with set attributes and global constraints as a unifying framework for planning domain models. In *Proc. of the 19th Int. Symp. on Principles and Practice of Declarative Programming (PPDP 2017)*, 39–48. ACM.

Barták, R.; Maillard, A.; and Cardoso, R. C. 2018. Validation of hierarchical plans via parsing of attribute grammars. In *Proc. of the 28th Int. Conf. on Autom. Plan. and Sched. (ICAPS 2018)*. AAAI Press.

Behnke, G.; Höller, D.; and Biundo, S. 2015. On the complexity of HTN plan verification and its implications for plan recognition. In *Proc. of the 25th Int. Conf. on Autom. Plan. and Sched. (ICAPS 2015)*, 25–33. AAAI Press.

Behnke, G.; Höller, D.; and Biundo, S. 2018. totSAT – Totally-ordered hierarchical planning through SAT. In *Proc. of the 32nd AAAI Conf. on AI (AAAI 2018)*, 6110–6118. AAAI Press.

Behnke, G.; Höller, D.; and Biundo, S. 2019a. Bringing order to chaos – A compact representation of partial order in SAT-based HTN planning. In *Proc. of the 33rd AAAI Conf. on AI (AAAI 2019)*. AAAI Press.

Behnke, G.; Höller, D.; and Biundo, S. 2019b. Finding optimal solutions in HTN planning – A SAT-based approach. In *Proc. of the 28th Int. Joint Conf. on AI (IJCAI 2019)*. IJCAI.

Bercher, P.; Alford, R.; and Höller, D. 2019. A survey on hierarchical planning – One abstract idea, many concrete realizations. In *Proc. of the 28th Int. Joint Conf. on AI (IJCAI 2019)*. IJCAI.

Bercher, P.; Behnke, G.; Höller, D.; and Biundo, S. 2017. An admissible HTN planning heuristic. In *Proc. of the 26th Int. Joint Conf. on AI (IJCAI 2017)*, 480–488. IJCAI.

Dix, J.; Kuter, U.; and Nau, D. 2003. Planning in answer set programming using ordered task decomposition. In *Proc. of the 26th Annual German Conf. on AI (KI 2003)*, 490–504. Springer.

Dvorak, F.; Bit-Monnot, A.; Ingrand, F.; and Ghallab, M. 2014. A flexible ANML actor and planner in robotics. In *Proc. of the 4th Work. on Plan. and Rob. (PlanRob 2014)*, 12–19.

Erol, K.; Hendler, J.; and Nau, D. 1994. UMCP: A sound and complete procedure for hierarchical task-network planning. In *Proc. of the 2nd Int. Conf. on AI Plan. Systems (AIPS)*, 249–254. AAAI Press.

Erol, K.; Hendler, J.; and Nau, D. 1996. Complexity results for HTN planning. *Annals of Mathematics and AI* 18(1):69–93.

Fox, M., and Long, D. 2003. PDDL2.1 : An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research (JAIR)* 20:61–124.

Geier, T., and Bercher, P. 2011. On the decidability of HTN planning with task insertion. In *Proc. of the 22nd Int. Joint Conf. on AI (IJCAI 2011)*, 1955–1961. AAAI Press.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research (JAIR)* 26:191–246.

Helmert, M. 2009. Concise finite-domain representations for pddl planning tasks. *Artificial Intelligence* 173(5-6):503–535.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2014. Language classification of hierarchical planning problems. In *Proc. of the 21st Europ. Conf. on AI (ECAI 2014)*, volume 263, 447–452. IOS Press.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2016. Assessing the expressivity of planning formalisms through the comparison to formal languages. In *Proc. of the 26th Int. Conf. on Autom. Plan. and Sched., (ICAPS 2016)*, 158–165. AAAI Press.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, B. 2018. A generic method to guide HTN progression search with classical heuristics. In *Proc. of the 28th Int. Conf. on Autom. Plan. and Sched. (ICAPS 2018)*, 114–122. AAAI Press.

Höller, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2019a. HDDL – a language to describe hierarchical planning problems. In *Proc. of the 2nd ICAPS Workshop on Hierarchical Planning*.

Höller, D.; Bercher, P.; Behnke, G.; and Biundo, S. 2019b. On guiding search in HTN planning with classical planning heuristics. In *Proc. of the 28th Int. Joint Conf. on AI (IJCAI 2019)*. IJCAI.

McDermott, D. 2000. The 1998 AI planning systems competition. *AI Magazine* 21(2):35–55.

Muise, C., and Lipovetzky, N. 2015. Unplannability IPC track. In *Proc. of the 2015 Works. on the IPC (WIPC 2015)*.

Nau, D.; Cao, Y.; Lotem, A.; and Munoz-Avila, H. 1999. SHOP: Simple hierarchical ordered planner. In *Proc. of the 16th Int. Joint Conf. on AI (IJCAI 1999)*, 968–973.

Nau, D.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, J.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research (JAIR)* 20:379–404.

Ramoul, A.; Pellier, D.; Fiorino, H.; and Pesty, S. 2017. Grounding of HTN planning domain. *International Journal on Artificial Intelligence Tools* 26(5):1–24.

Schreiber, D.; Balyo, T.; Pellier, D.; and Fiorino, H. 2019. Tree-REX: SAT-based tree exploration for efficient and high-quality HTN planning. In *Proc. of the 29th Int. Conf. on Autom. Plan. and Sched. (ICAPS 2019)*. AAAI Press.

Shivashankar, V.; Alford, R.; Kuter, U.; and Nau, D. S. 2013. Hierarchical goal networks and goal-driven autonomy: Going where AI planning meets goal reasoning. In *Goal Reasoning: Papers from the ACS Workshop*, 95–110.

Smith, D.; Frank, J.; and Cushing, W. 2008. The anml language. In *Proc. of the Work. on Knowledge Engineering for Plan. and Sched. (KEPS 2008)*.

Insights from the 2018 IPC Benchmarks

Isabel Cenamor and **Alberto Pozanco**
Departamento de Informática, Universidad Carlos III de Madrid
Avda. de la Universidad, 30. 28911 Leganés (Madrid). Spain
icenamorg@gmail.com, apozanco@pa.inf.uc3m.es

Abstract

The International Planning Competition (IPC) empirically evaluates state-of-the-art planning systems on a set of benchmark problems. The selection of this benchmarks plays an important role in the competition, since they can significantly affect competition results.
In this paper we analyze the diversity of the benchmarks employed in the last IPC through extracting some features from the domains and problems of the optimal track. Finally, we provide some insights from the collected data and propose to use a similar method to select the benchmarks of future competitions.

Introduction

In Artificial Intelligence, it is common to have competitions associated with each particular research area. These competitions aim to bring together different state-of-the-art systems, evaluating them on a set of benchmarks. Just like in Satisfiability Testing (Järvisalo et al. 2012), or in Answer Set Programming (Gebser, Maratea, and Ricca 2017), the Automated Planning community promotes the development of innovative planning techniques since 1998 through the International Planning Competition (IPC).

In the IPC, participating planning systems are tested in several benchmark problems. The selection of these benchmark domains and problems instances plays an important role in the competition, since they can significantly affect competition results (Howe and Dahlman 2002). This task is non-trivial, and it has given a lot of headaches to the organizers of previous competitions (Linares López, Celorrio, and Olaya 2015; Vallati, Chrpa, and McCluskey 2018). One of the main consensuses among the different post-competition discussions, is that benchmark domains and problems should be as diverse as possible (Vallati and Vaquero 2015), in order to (1) enrich the competition, and (2) not bias the results in favour of any planner.

In this paper, we analyze the diversity of benchmarks employed in the IPC 2018. We do that by extracting some features from the domains and problem instances of the optimal tracks. Features from domains and problems have been successfully used to predict planner's coverage (Roberts et al. 2008; Roberts and Howe 2009) or run time (Fawcett et al. 2014); and also to generate state-of-the-art planning portfolios (Cenamor, de la Rosa, and Fernández 2016). Here we use these features to evaluate and analyze the diversity of the competition benchmarks.

In the rest of the paper we introduce the feature extraction process, including a brief description of the features. Then, we detail how we process the raw data and introduce our different analyses, in which we also include the IPC 2014 for comparison purposes. Firstly, we perform an intra-domain analysis to test how diverse are the problem instances within the same planning domain. Secondly, we perform an inter-domain analysis to test how diverse are the domains and problems among them. Finally, we merge the data from IPC 2014 and IPC 2018 to group the domains and problems based on their similarity. We conclude our analysis by providing some insights from the results, and outlining a procedure similar to the one we carried out to select the benchmarks of future IPCs.

Planning Features

We use the same features extracted by the IBaCoP family of portfolios (de la Rosa, Cenamor, and Fernández 2017). The extraction process collects data from different steps of the Fast Downward system (Helmert 2006), in the version that was available before the IPC 2014. We briefly describe the set of 114 real-valued features we will use throughout our analysis by classifying them into the following categories:

- **PDDL**. These features are extracted from the original domain and problem definition in the PDDL files. If the domain contains conditional effects, we parse them using ADL2STRIPS (Hoffmann et al. 2006). Specifically, we have implemented the compilation that creates artificial actions for effect's evaluation (Nebel 2000). Some of these features are: number of actions, number of objects or number of goals.
- **Fast Downward Instantiation**. The pre-processor of Fast Downward instantiates and translates the planning tasks into a finite domain representation (Helmert 2009). Some of these features are: number of mutex groups, memory used for the translation process or whether action costs are used or not.
- **SAS$^+$**. These features are based on the causal graph (Helmert 2004) and domain transition graphs (Jonsson and Bäckström 1998) associated to the finite domain representation. Some of these features are: number of

variables and edges of the causal graph, ratio of variables involved in the goal, or sum of the number of nodes of all domain transition graphs.

- **Heuristics**. These features represent different heuristic values of the initial state of the search. Some of these features are: the FF heuristic (Hoffmann and Nebel 2001), the landmark-cut heuristic (Helmert and Domshlak 2009) or the red-black heuristic (Katz, Hoffmann, and Domshlak 2013).
- **Fact Balance**. These features are extracted from the relaxed plan of the initial state when the FF heuristic is computed.
- **Landmarks**. These features are extracted from the landmark graph computed by Fast Downward (see details in (Cenamor, de la Rosa, and Fernández 2016)). Some of these features are the number of landmarks, the number of edges in the landmark graph or the number of intermediate nodes in the graph.

Through extracting these features, we aim to characterize each problem instance to later compare them.

Data Extraction

We extract the IBaCoP features of the domains and problems of the IPC 2018 optimal track[1]. We decided to perform our analysis on that track given that we were able to successfully extract most of the features for all the problem instances, while the extraction results were worse in the satisficing track due to time and memory issues. We also extracted the features of the IPC 2014 optimal track[2] so we can compare them properly. The feature extraction process was run on an Intel Core i5-2410M CPU @ 2.30GHz and 4GB of RAM. We apply a time limit of 1800 seconds to the extraction of the features of each problem[3].

	#Features	Success
PDDL	8	100%
FD	16	100%
SAS$^+$	50	100%
Heuristic	16	82%
FB	10	76%
Landmarks	14	100%
Total	114	

Table 1: Feature type, number of features per type, and extraction success.

Table 1 shows the extraction success for each feature type in the IPC 2018. As we can see, most of the features are extracted correctly. Table 2 shows different metrics related to the time needed to extract the features in each domain of the IPC 2018. While it is easy to extract the features in some domains such as termes and data-network, there are other domains like spider or organic-synthesis-split in which this process may take up to two more orders of magnitude. This is because these domains present ADL, action costs, and negative preconditions, which need a special PDDL pre-process. We discarded organic-synthesis, since we only could extract the features of 5 problem instances within the time limit.

Name	Min	Max	Mean	Std	Median
agricola	32.0	164.0	87.9	36.2	80.0
caldera	97.0	382.0	319.8	76.6	339.0
caldera-split	64.0	280.0	135.3	50.0	124.0
data-network	2.0	6.0	3.6	0.9	3.0
nurikabe	61.0	634.0	311.1	169.1	324.5
organic-synthesis-split	80.0	**844.0**	257.8	**230.8**	133.0
organic-synthesis	-	-	-	-	-
settlers	100.0	283.0	177.6	52.1	167.5
snake	12.0	53.0	29.5	13.1	26.0
spider	**167.0**	671.0	**383.6**	137.3	**408.0**
termes	2.0	3.0	2.2	0.4	2.0
petri-net	7.0	30.0	18.4	6.8	17.5

Table 2: Minimum, maximum, average, standard deviation and median time to extract features for each domain in the IPC 2018 optimal track. In bold the higher values per column.

Data Pre-processing

After extracting the features, we have a features matrix $\mathcal{M}$. Each row in the matrix represents a problem instance p_k, and each column represents a feature f_i. Each cell contains the numeric value of a feature for that problem, $f_i(p_k)$. As we showed in Table 1, we do not have all the features' values for all the problems. So first of all, we have to deal with the missing values.

Here we have two main options: (1) discard those features with any missing value for any problem; or (2) substitute the missing values by actual values. Discarding features implies losing information. If the system is not able to extract a feature in a problem, it means that this instance is different from others in which it can be extracted. Moreover, we have high extraction success in almost all the features, so we opted for the second alternative, substituting the missing values. These values can be replaced in many ways. We chose to replace them by either:

- Setting the feature value to 0, if there is no problem in the domain in which the feature has been successfully extracted.
- Setting the feature value to the average of that feature values in the domain, if there exist at least one problem in the domain in which the feature has been successfully extracted.

Then, we cleaned the data by removing the features that were not sufficiently informative. For this purpose, we deleted the set of features which have the same value for all

[1] https://bitbucket.org/ipc2018-classical/domains/src

[2] https://helios.hud.ac.uk/scommv/IPC-14/benchmark.html

[3] The extracted data is available at https://github.com/apozanco/wipc-icaps2019 for IPC 2014 and 2018

the problem instances of the competition. This make our set of features to reduce from 114 to 107. We deleted 3 features from the PDDL description, 3 from the SAS^+ representation, and 1 from the heuristic values.

Finally, we normalized the features matrix by applying the following equation to every remaining feature $f_i \in \mathcal{M}$,

$$f'_i(p_k) = \frac{f_i(p_k) - f_{min}}{f_{max} - f_{min}}$$

where $f_i(p_k)$ is the current value of the feature f_i in the problem instance p_k; f_{min} and f_{max} are the minimum and maximum values of the feature f_i for all the problem instances $p_k \in \mathcal{M}$; and $f'_i(p_k)$ is the new normalized value of the feature f_i in the instance p_k. After this process, the features matrix $\mathcal{M}$ is normalized, with all the features' values within the $[0, 1]$ range.

If we take a look to these features' values, there are some that have similar values for all problem instances, while others are very different. As instance, in the IPC 2018, all the problem instances have a similar ratio between the total number of variables and the total number of edges in the causal graph. On the other hand, the most different features correspond with the number of predicates and types in the problem instances.

In the following experiments, we will refer as problem features' vector $\mathcal{V}_{p_k}$ to the list of values that describe the features of a problem instance p_k.

Intra-domain Analysis

Our first analysis aim to test how diverse are the problem instances within the same planning domain. For each planning domain, we compute a matrix with the columns and rows being the problem instances p_k of that domain. Each cell of the matrix denotes the difference between two problem features' vectors $\mathcal{V}_x$ and $\mathcal{V}_y$. This difference is computed as follows:

$$\mathcal{V}_x - \mathcal{V}_y = \sum_{i=1}^{i=107} |f_i(p_x) - f_i(p_y)|$$

This value can range from 0 to 107. Values closer to 0 mean that the two problem instances are similar, while higher values mean diverse problem instances.

To test how diverse the problems of a domain are, we then sum all the rows (or columns) in the matrix and divide that number by n^2, where n is the number of problems in the domain. This number reflects how different/similar is an average problem with the rest of problem instances of its domain. This number can range from 0 to 37.5 in the case of domains with 20 problems. Table 3 show the results of our intra-domain analysis for both IPC 2014 and 2018.

As we can see, there is a lot of variation in the results. Domains like parking and visitall in 2014, and spider or organic-synthesis in 2018 have very diverse problems, while all the instances in data-network or barman seem to be similar. The IPC 2018 has more diverse problems within the same domain, with an average difference of 5.6 against the average difference of 4.5 in the case of the IPC 2014.

Domain 2014	**Difference**	**Domain 2018**	**Difference**
parking	9.5	spider	12.5
tetris	8.5	organic-synth	10.5
visitall	7.4	nurikabe	9.9
transport	6.7	caldera	7.5
tidybot	4.9	caldera-split	4.4
openstacks	4.8	petri-net	4.1
citycar	4.1	agricola	4.1
cave-diving	4.1	snake	3.8
hiking	3.6	settlers	2.2
GED	3.5	termes	1.5
child-snack	2.2	data-network	0.9
floortile	1.4		
maintenance	1.1		
barman	0.9		

Table 3: Intra-domain Analysis.

To better illustrate how diverse the problems within a domain are, we plotted together all the problem features' vector of each domain. The results for the domains with most and least similar problems in the IPC 2018 are shown in Figure 1.

We also ran a small experiment to see if these results correlate with the planners' performance. We hypothesized that in domains with similar problems such as data-network, planners would perform similarly, i.e., they would solve most or almost none of the problems in the domain. On the other hand, in domains with different instances such as spider, planners would solve the problems in a more different way. To check this, we computed the standard deviation of each planner solving the problems of each domain (1 if a problem is solved, 0 otherwise). Lower values for a planner imply that it has been able to solve most or almost none of the instances in the domain. Then we compute the average of each planner for each domain.

However, our hypothesis is not met. As instance, spider which is the domain with most different instances, has a standard deviation of 0.47, while data-network which is the domain with most similar problems, has a standard deviation of 0.49. Some of the possible reasons why these results do not correlate are: (1) the competitor planners are very different from each other, and hence some domains and problems could be more suitable for one or other planner; and (2), the fact that a domain has similar problems does not necessarily imply that they can be solved in the same way. This little differences may come from increasing the number of objects, and therefore planners will only solve a small subset of them.

Inter-domain Analysis

Our second analysis aim to test how diverse are the domains among them. For each planning domain, we compute a domain features' vector $\mathcal{V}_{D_k}$ which is a problem features' vector representative of the domain D_k. We do that by assigning to each feature the average of the values of that feature in all the problem instances of the domain.

$$\forall f_i \in \mathcal{M}, f_i(\mathcal{V}_{D_k}) = \frac{\sum_{p_k \in D_k} f_i(p_k)}{|D_k|}$$

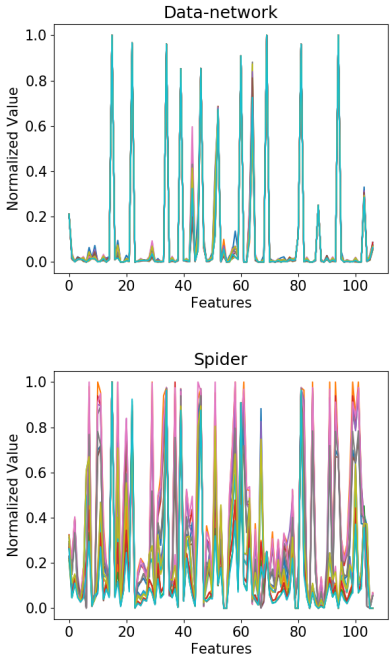


Figure 1: Domains with most similar (top) and least similar (bottom) problems. Each color represent one of the 20 different problem features' vector of each domain.

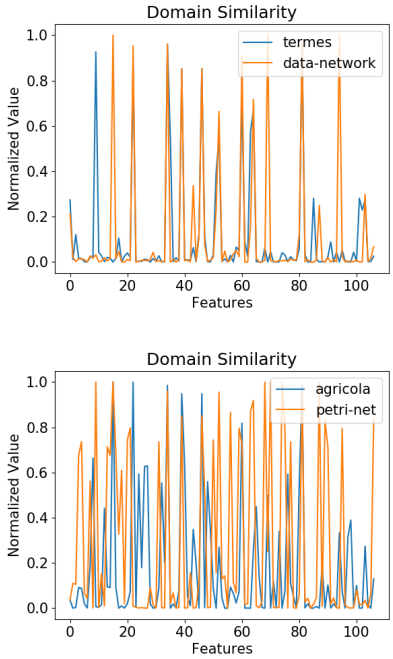


Figure 2: Similar domains (top) and different domains (bottom) in the IPC 2018.

By doing this, we are capturing all the information of a domain within just one features' vector. However, we may lose some information, mostly in those domains with diverse problem instances.

After that, we compute a matrix that in this case will have domain features' vectors both in the rows and columns. Each cell of this matrix denotes the difference between two domain features' vectors. This value can range from 0 to 107. If we sum each row (column) in the matrix and divide that number by k, the number of domains in the competition, we get how different is on average a domain with respect to the other domains. Table 4 shows the results of our inter-domain analysis of the IPC 2018.

As we can see, petri-net-alignment is the domain that keeps more differences with respect to the rest of domains, with an average value of 27.2. Moreover along with agricola, they are the most different pair of domains. On the other hand, caldera is the domain which is more similar to the others in the competition, with an average value of 12.0. Caldera and caldera-split is the most similar pair of domains in the IPC. This make sense, since both domains only differ in the problems' grounding. To better illustrate how diverse are the domains among them, we plotted together some domain features' vectors together in Figure 2.

We performed the same inter-domain analysis for the IPC 2014. In this case, the most different domain is tidybot, with an average value of 21.4. On the other hand, barman is the domain which is more similar to the others in the competition, with an average value of 9.2.

These maximum and minimum values are less distanced than the values of the IPC 2018. In fact, while the average of the domains' differences in the IPC 2018 is 15.6, this average is 12.0 in the case of the IPC 2014. This means that the set of domains and problem instances in the IPC 2018 is more diverse that the one of the IPC 2014.

We also ran a small experiment to see if these results correlate with the planners' performance. We hypothesized that planners would perform akin in similar domains and different in domains with different features. To check this, we computed for every planner the difference in absolute value of the number of problems solved in each pair of domains. Then we sum the results of each planner for each combination of domains and divide it by the number of planners. Lower values imply that the planners of the competition solve a similar number of problems in the given domains.

In this case, our hypothesis is met in most cases. As instance, if we take termes and data-network (the most similar domains except for the two versions of caldera), we get a value of 0.16, while in the case of agricola and petri-net (the most different domains), we get a value of 0.29. This is a common trend across domains, although there are some cases in which it is not fulfilled. As instance, the value obtained when comparing nurikabe and organic-synthesis is 0.13, which is lower than in the case of termes and data-network. Again, this can happen for the same reasons described in the intra-domain analysis.

	organic-synths	agricola	caldera-split	spider	termes	data-network	snake	nurikabe	caldera	petri-net	settlers
organic-synth	0.0	19.4	14.2	20.4	17.6	15.6	16.9	15.3	13.0	33.3	11.9
agricola	19.4	0.0	16.7	22.9	18.1	13.8	17.2	15.5	14.4	33.5	15.3
caldera-split	14.2	16.7	0.0	16.8	10.3	10.7	13.4	12.9	7.7	26.9	8.3
spider	20.4	22.9	16.8	0.0	20.7	22.1	16.2	19.3	15.7	33.2	18.3
termes	17.6	18.1	10.3	20.7	0.0	8.3	11.1	14.0	11.5	25.0	12.4
data-network	15.6	13.8	10.7	22.1	8.3	0.0	13.9	13.1	10.6	25.8	10.2
snake	16.9	17.2	13.4	16.2	11.1	13.9	0.0	13.1	10.4	31.7	13.7
nurikabe	15.3	15.5	12.9	19.3	14.0	13.1	13.1	0.0	10.7	31.3	10.8
caldera	13.0	14.4	7.7	15.7	11.5	10.6	10.4	10.7	0.0	29.2	8.9
petri-net	33.3	33.5	26.9	33.2	25.0	25.8	31.7	31.3	29.2	0.0	28.9
settlers	11.9	15.3	8.3	18.3	12.4	10.2	13.7	10.8	8.9	28.9	0.0
Average	16.1	17.0	12.5	18.7	13.5	13.1	14.3	14.2	**12.0**	**27.2**	12.6

Table 4: Differences among domains from the IPC 2018. Green cells identify diverse domains, while purple cells identify similar domains. Bold numbers represent the most diverse and similar domains in the competition.

Clustering Domains

Our last analysis aim to group the benchmarks of the IPCs 2014 and 2018 based on their similarity. For this purpose, we merge the raw data of the extracted feaures of both competitions, and follow the same pre-processing step as before. We compute a features' vector for each domain, as we did in our inter-domain analysis.

Now we perform a hierarchical clustering to the 25 domain features' vectors (11 from the IPC 2018 and 14 from the IPC 2014). We do that to test (1) if there exist similar domains across different competitions, hence being part of the same cluster; and (2) which domains are the most different from the rest, hence conforming they own cluster.

Figure 3 shows the result of our hierarchical clustering in the shape of a dendrogram. As we can see, domains like barman, child-snack or hiking are grouped together first. This means that they are the most similar ones within both competitions. The most diverse domains are shown at the bottom of the y axis. They correspond to spider, settlers, visitall, agricola, parking, tidybot, organic-synthesis and petri-net-alignment, which is the most different domain across competitions.

Discussion

The selection of the benchmark domains and problem instances plays an important role in the IPC. A desirable property of these benchmarks is that they should be as diverse as possible, in order to enrich the competition and not bias the results in favor of any planner. In this paper we have presented a study of the diversity of the benchmarks of the IPCs 2018 and 2014. We carried out three different analyses: an intra-domain analysis, to test how diverse are the problem instances within the same planning domain; an inter-domain analysis, to test how diverse are the domains and problems among them; and a clustering procedure to group the domains of both IPCs based on their similarity.

Our analyses suggest that the IPC 2018 employed more diverse domains and problem instances than the IPC 2014. From the results, we can also conclude that in both competitions there are domains which are not similar to any other, not only within the same competition but also if we take other IPCs into account. We think these *different* domains such as spider, agricola or parking really enrich the IPC.

However, our results should be read carefully, and more like a photograph of the benchmarks, than a test that determines how good or bad a problem/domain/competition is.

The first reason for that is that throughout our analyses, we measure the similarity or diversity of problems and domains with respect to their set of features. These features, even though proved useful by other works, may not conform the best set of features for differentiating problems; also, some of these features may be too correlated and introduce noise in the similarity computation. Further work on the set of features should be done to properly characterize problem instances. We also want to note that the fact that a domain has similar problem instances, or a competition similar domains, does not mean anything bad. It may be the case that all these similar domains are challenging for the planners. Moreover, the low intra-domain differences in domains like barman may be related to having problems with increasing number of objects or goals. These type of domains are useful to test planners' scalability and should be present at future competitions.

The second reason is that a competition comprises both benchmarks and planners. Although other works has focused on that relationship (Cenamor, de la Rosa, and Fernández 2016; de la Rosa, Cenamor, and Fernández 2017), here we only focused on the benchmarks, leaving the planners' performance over these benchmarks out of the scope of this paper. This work should be extended to take diverse planners into account, characterizing them and analyzing how they solve each kind of domain and/or problem instances. By doing this, it would be possible to know which set of features make the problem instances hard to solve by each kind of planner. This information would be very useful when selecting the domains and problems of a competition.

We believe that by improving this work in the outlined directions, we may have some of the key ingredients to select (or even generate) diverse benchmarks for future IPCs.

Acknowledgments

Alberto Pozanco is funded by FEDER/Ministerio de Ciencia, Innovación y Universidades Agencia Estatal de Investi-

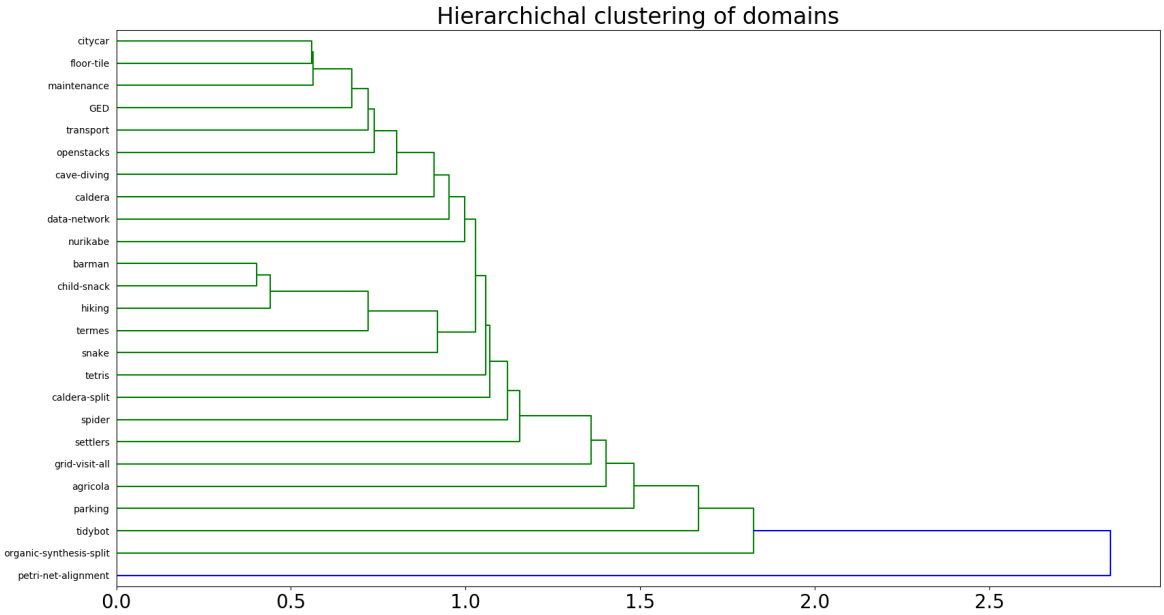


Figure 3: Hierarchical clustering of domains. The domains are represented in the y axis, while the x axis represents a measure of error. Domains grouped first are the most similar. Domains grouped last, depicted at the bottom of the y axis, are the most different.

gación/TIN2017-88476-C2-2-R and RTC-2016-5407-4.

References

Cenamor, I.; de la Rosa, T.; and Fernández, F. 2016. The IBaCoP planning system: Instance-based configured portfolios. *J. Artif. Intell. Res.* 56:657–691.

de la Rosa, T.; Cenamor, I.; and Fernández, F. 2017. Performance modelling of planners from homogeneous problem sets. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017.*, 425–433.

Fawcett, C.; Vallati, M.; Hutter, F.; Hoffmann, J.; Hoos, H. H.; and Leyton-Brown, K. 2014. Improved features for runtime prediction of domain-independent planners. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*.

Gebser, M.; Maratea, M.; and Ricca, F. 2017. The sixth answer set programming competition. *Journal of Artificial Intelligence Research* 60:41–95.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004), June 3-7 2004, Whistler, British Columbia, Canada*, 161–170.

Helmert, M. 2006. The fast downward planning system. *J. Artif. Intell. Res.* 26:191–246.

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. *Artif. Intell.* 173(5-6):503–535.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res.* 14:253–302.

Hoffmann, J.; Edelkamp, S.; Thiébaux, S.; Englert, R.; dos S. Liporace, F.; and Trüg, S. 2006. Engineering benchmarks for planning: the domains used in the deterministic part of IPC-4. *J. Artif. Intell. Res.* 26:453–541.

Howe, A. E., and Dahlman, E. 2002. A critical assessment of benchmark comparison in planning. *J. Artif. Intell. Res.* 17:1–3.

Järvisalo, M.; Le Berre, D.; Roussel, O.; and Simon, L. 2012. The international SAT solver competitions. *AI Magazine* 33(1):89–92.

Jonsson, P., and Bäckström, C. 1998. State-variable planning under structural restrictions: Algorithms and complexity. *Artif. Intell.* 100(1-2):125–176.

Katz, M.; Hoffmann, J.; and Domshlak, C. 2013. Red-black relaxed plan heuristics. In *Proceedings of the Twenty-*

Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA.

Linares López, C.; Celorrio, S. J.; and Olaya, A. G. 2015. The deterministic part of the seventh international planning competition. *Artif. Intell.* 223:82–119.

Nebel, B. 2000. On the compilability and expressive power of propositional planning formalisms. *J. Artif. Intell. Res.* 12:271–315.

Roberts, M., and Howe, A. E. 2009. Learning from planner performance. *Artif. Intell.* 173(5-6):536–561.

Roberts, M.; Howe, A. E.; Wilson, B.; and desJardins, M. 2008. What makes planners predictable? In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008*, 288–295.

Vallati, M., and Vaquero, T. 2015. Towards a protocol for benchmark selection in IPC. *In Proceedings of the 4th Workshop on the IPC*.

Vallati, M.; Chrpa, L.; and McCluskey, T. L. 2018. What you always wanted to know about the deterministic part of the international planning competition (IPC) 2014 (but were too afraid to ask). *Knowledge Eng. Review* 33:e3.

Cost-Optimal Planning in the IPC 2018: Symbolic Search and Planning Pattern Databases vs. Portfolio Planning

Stefan Edelkamp and **Ionut Moraru**
King's College London
{stefan.edelkamp,ionut.moraru}@kcl.ac.uk

Abstract

The optimal track is one the most exiting events in the International Planning Competition (IPC).

In this position paper we argue that —despite of not winning the competition— symbolic search and pattern databases were likely the most influential planning approaches in the latest IPC in 2018, and, in continuation to the precursor IPC in 2014, should be considered as candidates for the current state-of-the-art.

Five of the Top 6 planners in the 2018 competition, namely Complementary (1 and 2), Planning-PDBs, Symbolic-Bi-directional, and Scorpion are based on these technologies. These planners use the same technology across all domains and plan in one state space.

The winner of IPC 2018 with an $\approx 1\%$ lead in problems being solved, however, is a so-called *portfolio* planner, consisting of a selection of many different planners, one of which is chosen in a classifier that was trained on a manually selected set of benchmark instances. In about half of its successful runs, it called the winner of the previous IPC, which in turn is based on symbolic search. We argue on whether or not to exclude portfolios from the IPC is possible and wanted.

Introduction

Being inspired by the ultimate goal of general problem solving, in the field of action planning, there is a common understanding that planning competitions ran on a set of unknown and partly new set of benchmark problems, advance planning technology the most.

Starting in 1998, over the years considerable progress has been made in the development of new planners, due to competitors being brave enough to face a coding contest with an unknown benchmark set and with the obligation to offer their source code. Competitors have been confronted with an increasing set of challenges including extended expressiveness of planning domain description language (PDDL), involved planning task metrics, in inherent problem complexity, and in scaling problem sizes. While several tracks were spawned, one is the deterministic part of the IPC, and its track on cost-optimal planning.

The outcome of the optimal track in the most recent 2018 International Planning is revisited in Table 1[1]. The corresponding output on IPC 2014 has been provided and discussed by (Edelkamp, Kissmann, and Torralba 2015).

While most planners present one sole planning technology, the winning planner Delfi (Sievers et al. 2019) is a *portfolio*, a mixture of different technologies. Given a problem task, it selects a planner based on a classifier, being trained on a manually chosen set of known planning benchmark instances. In its set of planners, 16 were contained as part of the Fast Downward planning framework, including at least one (Canonical PDB) that used pattern databases. The most effective approach chosen by this classifier, however, was the symbolic search planning system, which won the precursor 2014 IPC competition. Moreover, in the only domain, where Delfi scored overall clear best, it called this planner.

From the mere planning side, in portfolios there is often little that is novel, the contribution of these planners is often found in the machine learning algorithm, which is trained on a set of known planning problems and which eventually selects the planner configuration to call. In Delfi one planner is called per instance. In terms of the potential of different planning approaches available to Delfi, the IPC outcome with a lead of two more being solved ($\approx 1\%$ of 200 benchmark problems) is quite small, showing that cost-optimal planning is tough, even for portfolios. A change in one domain would have resulted a different outcome.

While portfolio planning was in alignment with the rules of the competition, one underlying issue is some participating planners avoided using portfolio technology, presumably in favour of getting a clearer picture on what technology is currently leading. They seemed to prefer working on new plan search technologies, instead of going in for a mixture of existing planners.

Facing the outcome of the competition and the different type of contributions available in portfolio and non-portfolio planners, people interested in planning especially outside to the core planning community have to be warned not to derive wrong scientific conclusions by only looking at the outcome. Competition results always have to be dealt with care.

This position paper aims to provide a clearer picture on what is the currently leading technology according to IPC 2018 and discusses on whether or not a portfolios help to push or blur the outcome of a competition. The stress of this

[1]Readers interested in the planners listed are forwarded to the according planner abstracts found in the IPC 2018 competition booklet at https://ipc2018-classical.bitbucket.io/planners

	Agri-cola	Cal dera	Data Net Network	Nuri-kabe	Organic Synthesis	Petri Net Alignment	Sett-lers	Snake	Spider	Termes	Σ
Delfi1	12	13	13	12	13	20	9	11	11	12	126
Complementary2	6	12	12	12	13	18	9	14	12	16	124
Complementary1	10	11	14	13	13	17	8	11	11	16	124
Planning-PDBs	6	12	14	11	13	18	8	13	11	16	122
SymbBiDir	15	10	13	11	13	19	8	4	7	18	118
Scorpion	2	12	14	13	13	0	10	14	17	14	109
Delfi2	11	11	13	11	13	9	8	7	7	15	105
FDMS2	14	12	9	12	13	2	8	11	11	12	104
FDMS1	9	12	10	12	13	2	9	11	11	12	101
DecStar	0	8	14	11	14	8	8	11	13	12	99
Metis1	0	13	12	12	14	9	9	7	11	6	93
MSP	7	8	13	8	12	10	0	11	6	16	91
Metis2	0	15	12	12	14	9	0	7	12	6	87
ExplBlind	0	8	7	11	10	7	8	12	11	10	84
Symple-2	1	8	9	7	13	2	0	0	5	13	58
Symple-1	0	8	9	8	13	2	0	0	4	13	57
maplan-2	2	2	9	0	6	0	0	14	1	12	46
maplan-1	0	2	12	0	6	0	0	7	10	6	43

Table 1: IPC 2018 results, measured in the coverage of benchmark problems, i.e, in the number of tasks solved per domains.

paper, therefore, is to argue that, while not winning the competition because of portfolio planning, the two technologies of BDD-based symbolic search planning (Edelkamp and Helmert 2001) and planning pattern databases (Edelkamp 2001) first joint up by (Edelkamp 2005) seemingly dominated the overall outcome competition.

Symbolic Search and Pattern Databases

The IPC 2018 planner Symbolic-Bidirectional (SymbBiDir) was suggested by the authors and accepted by the organizers as a baseline planning technology. It includes no lower bound at all and, thus, relies on so-called *blind* search, i.e., a search with no heuristic search guidance. As actions carry cost, instead of a breadth-first exploration this induces a cost-first traversal of the state-space graph.

In *symbolic planning*, the core difference to explicit-state space planning is the use of binary decision diagrams (BDDs) to represent state sets in the search compactly (Bryant 1986). As actions can also be represented in form of BDDs encoding the transition relation, it is possible to progress and regress planning state in this succinct functional state set representation to perform forward and backward exploration in an operation called *relational product* (Clarke et al. 1996). A first A*-type algorithm for BDD-based heuristic search has been proposed by (Edelkamp and Reffel 1998).

One obsevation is that memory savings obtained via the compact representation in a BDD in turn often also lead to significant savings in CPU time. The gain of a symbolic representation in IPC 2018 is amplified, when comparing the performance gap of SymBiDir with the other baseline planner ExplicitBlind. As the names indicate, the two baseline planners are not executing the same exploration, due to the fact that coding regression search is not immediate for the usually given partial goal representation; so that the latter conducts a forward state-space traversal only.

To the contrary, SymBiDir executes bidirectional cost-based search, much in the sense of bidirectional application of Dijkstra's single-source shortest path algorithms (Dijkstra 1959), taking care of the fact that the optimal solution might not be established on the first meeting of both search frontiers. As the BDDs represent state sets, recursive solution construction is needed for extracting optimal plans. Aspects like a partitioned computation of successors (called the *image*), variable ordering based, as well as the inclusion invariant constraints to rule out illegal and dead-end states turn out to be crucial factors to improve the exploration efficiency (Torralba et al. 2017).

The performance results of SymBiDir revealed, that in only two of the ten domains (Snake, Spider) is was not doing well, otherwise the baseline planner would have won the competition! This indicates the power of symbolic state space representation and exploration, and suggests that at least across the entire IPC 2018 benchmark set, the vast amount of refined heuristics for planning do not always lead to a throughout leading technology.

According to the result of the 2018 competition, planning pattern databases (PDBs) (Edelkamp 2001) appears to be one of the few exceptions. On the testbet of IPC 2018 the combination of PDBs and symbolic search in the planners Complementary (1/2) and Planning-PDB are outperforming Symbolic Bidirectional Search. The former two are inspired by results of (Franco et al. 2017), while the latter improves on bin packing algorithms for the pattern selection problem. Besides a major rewrite, one new feature of these new planners is that the forward search is in fact explicit-state, while only the backward traversal is symbolic.

Of course, many heuristics besides PDBs are still worth further investigation. For the case if Snake and Spider were removed, it is difficult to draw a conclusion on different

types of heuristics from the IPC-18 results.

There is no free lunch. But the overall performance of bidirectional symbolic search is surprisingly good, while not using any heuristic. Whether the aspect of bidirectional or the symbolic search contributes the most to this performance, we haven't checked, but we expect one would need both.

SymbBidir performs much better in Agricola than the other planners. Using a PDB heuristics seems to hurt symbolic search and fails in 5 to 9 instances where SymbBiDir succeeds. Some of these problems have been identified and overcome by using *perimeter pattern databases* (Felner and Ofek 2007).

About the bad performance in Spider and Snake there are reasons on why the BDD *explode*, which relate to the subtle ordering problem of dependent BDD variables in grids, This issue has been analyzed and proven to be crucial for representing the goal to ConnectFour as a BDD (Edelkamp and Kissmann 2011), and might be detected fully automatically.

Recall, that planning pattern databases are serving as heuristics and are based on a complete backward exploration in some state-space abstraction. Often a larger number of (hopefully) diverse and complementary patterns are generated and the corresponding databases sought to be combined in an admissible manner to preserve being a lower bound.

While Scorpion is partly a PDB planner performing slightly worse to SymbBiDir in IPC 2018, it showed distinguished performance and scored best in 5 out of 10 domains. Further investigations illustrate that it is performing much better across all IPC benchmarks, i.e., ones including the ones from previous competitions (Seipp and Helmert 2018; 2019). It also has to be added that part of the success of Scorpion is due to Cartesian abstractions combined with a counterexample abstraction refinement (CEGAR) approach (Clarke et al. 2000).

As both heuristics are based on state space abstractions, one may view PDB and Cartesian abstraction heuristics as being related estimates of a similar type. While PDBs and Cartesian abstractions are an important part of Scorpion, much of its strength is in the sophisticated method to *combine* these abstraction heuristics. While the competing BDD based PDB planners mainly use 0/1 cost partitioning a more advanced concept saturated cost-partitioning.

When normalizing the success alongside the domains (some have 150 instances), Scorpion compares well with the top-tier symbolic planning systems: 0.621 (Complementary 2), 0.601 (Scorpion), 0.594 (Planning PDBs), 0.576 (Complementary 1), 0.555 (SymBiDir). Using Delfi in this comparison hardly applies, as for this case the training set overlaps the test set.

Again, all or most planners, even the ones that were not at the top, had some positive results (e.g. being among the planners that solve most instances in some individual domains). Even planners at the bottom have some cases where they perform among the best (e.g. ma-plan in Snake).

Portfolio Planning

Portfolios erected on existing plan technology, are a recurring pattern in many competitions, and range from restarting strategies, over learning classifiers, to scheduling time slices to existing planners.

Once having fixed the metric and submission instructions, the IPC 2018 organizers felt that they had to follow them. If a planner wins by the metric they decided on before, the competition and the organizers didn't crown it the winner, the teams would rightfully complain.

The coverage metric of problems being solved seemed not to be the issue, so one thing the organizers could have done, would have been to change the rules for submissions. They explained that they had an internal discussion about portfolios early on in the course of running the competition, but decided that the line between a planner with multiple components and a portfolio was too blurry to accurately define. That is why they decided to not have a special rule against portfolios. As one reason given, the LAMA system (Richter and Westphal 2010), one previous IPC winner, might have to be considered a portfolio, because it runs different planners one after the other.

In terms of the organizers of the competition is difficult to set rules that identify portfolios to give them a special treatment, because either they'd be too restrictive and most planners of the competition would be considered portfolios, or they would be too ambiguous, generating complains about what planners are considered to be portfolios. This said, there are certainly many interesting aspects to be learned from portfolios on a per-domain or even per-instance base. Proper portfolio designs with a close-to-optimal choice of planners as in Delfi is a research area on its own.

One way to limit the impact of portfolios in the competition is what Mauro and others have suggested for the Sparkle planning competition[2], where planners are evaluated based on how well they do on individual instances/domains, rather than on getting a good average score. However, this suggestion comes with some issues as well. In particular, the score of a planner completely depends on which other planners are submitted to the competition. Henceforth, if someone submits a version of your planner that works only slightly better, he could get 0 points. One has to wait for the results to see how the approach materializes. Unfortunately, the organizers of this competition are only running the agile track, without insisting on cost-optimal plans. In complexity terms, however, optimality is known to be of crucial importance. For example, finding any plan for many planning benchmarks (such as Blocksworld, Logistics, Sliding-Tile Puzzle) is polynomial. In this case of *satisficing* planning where only plan existence is requested problem tend to be tractable, while the corresponding optimization are often provably hard (Slaney and Thiébaux 2001; Parberry 2015; Helmert 2008).

The emerging set of portfolio and the difficulty of excluding them may be seen as a side effect of the requirement of releasing source code for the planners, as it becomes easier to bundle the planners into one code base. Of course public access to the source code is not a strict necessity for these type of planners.

For some planning researchers, the core issue and concern

[2] http://ada.liacs.nl/events/sparkle-planning-19

of portfolio planning is that other researchers use their code, and not so much that the participating planner is a portfolio. So the organizers thought about having had a rule against using code from another research groups. That would have excluded most planners based on Fast Downward planner framework, though, and since they were the majority of the submissions, this would not have been a good idea as well.

The solution Fahiem Bacchus suggested (in personal conversation with the authors) based on his own experience with portfolios in the SAT competitions, is having stated a license that prohibits the use of the code in other tools, would be an option, but then the authors of the planners would have to do so before the competition. In case of planners based on Fast Downward, this, however, is also problematic because such a clause would be hardly compatible with the license of the framework.

While intuitively rather obvious, it is far from simple to distinguish portfolio from non-portfolio planner in a formal definition. One may try to start with the following criteria.

> A *planner portfolio* selects, invokes, and possibly terminates different existing planners, based on a trained or hard-coded decision procedure.

This definition may not be a perfect discriminator, as one might be able to transform a portfolio into a non-portfolio without changing the performance by much: just moving the decision procedure further down the line.

It does also not cover a planner that uses the maximum of ten heuristics in an A* search. Some people would like to treat this as a portfolio, because there is no contribution except for the selection procedure of the ten heuristics. At the end, the question remains on when a planner is a novel contribution.

Another suggested definition for identifying portfolio planning is the following.

> A *non-portfolio planner* is a single core planning technology, which invokes a plan search in one state space.

But what is with traversing state-space abstractions, which are needed to compute heuristic estimates? Clearly, as highlighted by the IPC organizers, defining portfolios turns out to be intrinsically difficult. There are planners that are clearly portfolios, there are planners that are clearly not, and there is a larger gray area in between.

According to a definition, FF (Hoffmann and Nebel 2001) should not be judged as a portfolio planner. It searches one state space with one heuristic. But FF switches from enforced hill-climbing to best-first search, based on some progress measure. This alone should not classify it as a portfolio approach.

LAMA runs a greedy search based on h^{FF} and a landmark heuristic (three techniques developed by different authors) and then several weighted A* searches (a different planner and an algorithm also developed by other authors). LAMA may or may not be seen as a portfolio. If it runs three independent searches in parallel, then this may be interpreted as a portfolio technology, but the interconnection of the search is more subtle. LAMA had additional algorithmic contributions on how to move back and forth the states in the different priority queues. If LAMA continues searching the same search space, this is a sign of a non-portfolio. At least it does not start different existing planner.

It is, however, abundantly clear that Delfi is a portfolio planner (not even the authors questions that). It even logs its task-dependent calls to the planner binaries. Delfi actually uses 2 executables, SymBA*, and 16 parameters of selecting planners in the Fast Downward framework. It has a decision procedure trained on a set of manual selected planning tasks. Note that in this setting, we do not count the learning as running, but as programming time.

The performance overhead of portfolio designs can be small. The often criticized effect is that frequently more than 99.9% of the actual running time of a portfolio planner is exclusively spend on existing technology. This is a probably unwanted aspect, which can makes other competitors that contribute non-portfolio planners wondering and reluctant to tune their planners for efficiency. Of course, the size of a contribution must not necessarily be taken in direct correspondence to the profiled time that was spend in running the code added.

By public access to the planners at IPC 2018, one can look at the source code of the contributed planner to validate, on whether or not a planner is in fact a portfolio.

Fast Downward's code base has grown too big, there are pros and cons to that. On the pro side the planner suite is good for benchmarking. For the symbolic search engines in IPC 2018, it was better to use it to combine explicit with symbolic search than sticking to an independent technology in Gamer. In fact, there are myriads of parameters that make Fast Downward behave totally different. Fast Downward is no longer one planner, it is a framework. On the cons side, results on mixing different calls it may blur the messages you to take home.

There were at least two different portfolio planners in IPC 2018: Delfi1 and Delfi2, where Delfi1 was so much better, so that in the following we concentrate on this one, and used Delfi for its shortcut notation. There is published work of the planner authors in the IPC booklet that explain the architecture and the machine learning approach of using deep neural nets in more detail, so that we concentrate on the main aspects. The main idea is to train a classifier on the performance curves of known planning benchmark problems, provided as input images. We had some problems to reproduce the results on our machine, but could look at the competition results. The planners being invoked by Delfi in the IPC 2018 are shown in Table 2. Note that Delfi combines the heuristics listed with symmetry and partial-order reduction.

The story on portfolios will go on. Portfolios have already started integrating the systems from 2018. Essentially, even when pushing the field with new and brilliant ideas, one hardly can win the race against a portfolio, at least in general terms.

While portfolios have dominated some tracks in the IPC (the satisficing track, for example) in the optimal track, the winner of IPC 2014 was not a portfolio. As seen in IPC 2018, there were several planners that got a very close performance to Delfi. Also, some other portfolios participated. Delfi2 and other portfolios (MSP and DecStar could be considered port-

Approach	Used	Successfully
SymBA*	110	73
LM cut	64	37
Merge&Shrink	47	20
Canonical PDB	17	13
Blind search	2	2
Total	240	147

Table 2: Planners chosen by the Portfolio Delfi1 based on analysing the log files of IPC 2018. Only main planner technologies are mentioned, many more parameters apply to the actual invocation of the code. Note that the number of problems being solved is slightly higher than in the competition outcome, as there were some reformulations of the same problem, where the planner was run, too.

folios as well) were behind many non-portfolio planners. Overall, the results do not show dominance of portfolio planners in general.

What is worse, with winning of the IPC in the pocket, portfolio planners help to acquire project money and to publish in high-ranked journals and conference proceedings, where non-portfolio planners often have a harder time arguing that they are carrying the actual contribution in technology, as especially in research, a second place is often not considered state-of-the-art.

Instead of arguing, whether portfolios should participate or not, we should discuss about how people interpret the results of the IPC and how an analysis that goes beyond *Planner X is the winner* is absolutely necessary. Even planners that are not at the top show that there are domains where they can be really useful. We view this position paper as one step towards this end.

About credits. For a scientific paper it is rather clear that one has to become a co-author, if one contributes substantially to the outcome. With portfolios this is slightly different. In the extreme case, it may happen tbat the one coder contributes and the other one coder using portfolio technology take the credits for the efficiencies of the work.

Of course we wouldn't ask Hart, Nilsson and Raphael to be co-authors of every forward-search planning paper using A* but we would still cite their work (Hart, Nilsson, and Raphael 1972). Probably the same is true for portfolio planners: they should give credit, where it is due (and the planner abstracts do this).

No question, portfolios also have their own contribution. The contribution in a portfolio planner is the combination of techniques, e.g., how much better is this combination than just running all n components for 1 n-th of the time.

In this respect the competition booklet helps a lot as it links the IPC planners to the outside people, but one the other hand, in the scientific race a booklet is no archival publication is rarely counted as a success. This is what one may ask for a portfolio, to be explicit on which planner call achieves which individual performance, and not to bury this information in a lot of other stuff, e.g., on how advanced the machine learning (e.g., deep neural network training) is.

We often insist on a proper publication before the release of the code, but this also does not work for competitions like IPC 2018. The problem is essential, as with the competition the coders provide all source to the public, so we should take more care on who contributes what.

If an outside contribution is dominating the own one consider asking the authors. Sometimes you cite and acknowledge, sometimes you feel this is authorship. For the IPC, we see people taking code, shake it a bit to improve the results slightly, and go on publishing.

Whether or not portfolio planners being trained on sample plans are domain-independent, is also a controversy, especially given that training plan samples selected by hand. Extremist think they are not, but other people may think differently. Surely portfolio planners belong to the learning track. Of course, organizers were quite happy to have that many competitors, and for us it was a tremendous success to see how good our planners performs, even when facing portfolios. We enjoyed to see how hard it is to get some good result in cost-optimal by machine learning.

Conclusion

This position paper aims at arguing on what is the currently leading technology and discusses whether or not portfolios help to push or blur the outcome of a competition. There is indication but not one definite conclusion alias strong proposal along this paper on how to deal with the given observations, the main purpose of the paper is look behind the scenes and to spawn a discussion. Such discussions have tradition in the IPC. We had a discussion on domain-independence at the emerge of control rules in TL- and TAL-Plan (Bacchus and Kabanza 2000; Kvarnström and Magnusson 2003), we also had a discussion on the effect of hand-coded selection in planners like SG-Plan (Wah and Chen 2004), with complaints on hand-written domain-dependent branching inside the planners' code. Interestingly, the one who complains most about the current IPC organizes the next IPC. Now in 2018 the topic is portfolio planning and the problem of identifying and securing individual planner contributions.

The international planning competition 2018 pushed the field in action planning, set up and executed a well-designed externally controlled experiment, aimed at insights about the true performance of planners, falsified and strengthened hypotheses on essential components, compared different technologies on a common rule set, same architecture, and an agreed input formalism. It awarded scientific prizes and provided opportunities for upcoming publications. The evaluation is much better than what one experiences in conference and journal papers. The results are often surprising, when compared to the wisdom taken from existing publications. Of course, every competition is limited in what it can prove, but its scientific impact is not to be underestimated.

This paper discusses two separate claims: 1) An analysis of the result of the optimal track, claiming that symbolic search and PDBs are leading methods for cost-optimal planning. 2) The advantages and disadvantages of considering portfolios as part of the IPC.

The IPC has always been a competition where the best mix of scientific and engineering skill wins. It has never

been just one good idea that won the competition, but also the skill to implement it efficiently, and even before portfolios there was a chance that the better software developer outperforms the better researcher, possibly even with something that would never be published.

In this case most arguments are made only about IPC 2018 (Delfi vs Complementary/PlanningPDBs). The discussion of what is the place of portfolios in the IPC and other competitions tracks and events is of course a a general one.

What to do? In a competition it is always good to refer to a wider set of planners, but fundamental differences should be highlighted and could have been put into the awarding considerations. It is fine to have portfolios inside the competition, but they should at least be tagged as such, given a portfolio is a different type of planner, and, otherwise, wrong conclusions might be drawn from the event. Otherwise, the competition is doomed to swallow its own core contributors. The risk is that these efforts will die out.

Considering on how many different planning heuristics have been suggested in the past, given that places two to six are symbolic search and planning pattern database planners only, and that the winner of IPC 2018 called a symbolic search planner half of its time is a striking fact. We are not aware of any technology that performs better especially on the 2018 IPC benchmark set. Given fluctuations in the results many planners are playing in the same ballpark.

Recent improvements and simplifications indicate that one can lift the results of symbolic pattern database planning towards winning the IPC 2018 competition post mortem.

While portfolios usually dominate non-optimal IPC tracks, it is indeed a tighter race in the cost-optimal track. Optimal planning seems to be a tough nut to crack for portfolios given the limits in time and space, even when having a bigger toolbox. Portfolios can take on novel contribution for free and quite quickly. The advance one to put on top from event to event is counterbalanced with using many planners at once. Whether such an advance is always possible, is a subject projection. But at least it may be argued that it appears that it might be exhausting for the competitors with non-portfolio planners to come up with novel, original and breakthrough technology at every new IPC and compete with the portfolios of the last one.

Acknowledgement Programming is a serious art and comes with a lot of fun. The IPC 2018 is a programming contest that allowed all competitors to impress with stunning and outstanding performance results on yet unseen complex, and diverse problem domains. We thank all competitors for the variety of new planning approaches, advancing the state-of-the-art in many respects. The organizers of the IPC 2018 did a great job, both with the choice/design of the benchmark domains and for running the competition. Eventually, they had to decide on the winner according to the rules set.

References

Bacchus, F., and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artif. Intell.* 116(1-2):123–191.

Bryant, R. E. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers* 35(8):677–691.

Clarke, E. M.; McMillan, K. L.; Campos, S. V. A.; and Hartonas-Garmhausen, V. 1996. Symbolic model checking. In *Computer Aided Verification, 8th International Conference, CAV '96, New Brunswick, NJ, USA, July 31 - August 3, 1996, Proceedings*, 419–427.

Clarke, E. M.; Grumberg, O.; Jha, S.; Lu, Y.; and Veith, H. 2000. Counterexample-guided abstraction refinement. In *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, 154–169.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271.

Edelkamp, S., and Helmert, M. 2001. MIPS: the model-checking integrated planning system. *AI Magazine* 22(3):67–72.

Edelkamp, S., and Kissmann, P. 2011. On the complexity of BDDs for state space search: A case study in Connect Four. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*.

Edelkamp, S., and Reffel, F. 1998. OBDDs in heuristic search. In *KI-98: Advances in Artificial Intelligence, 22nd Annual German Conference on Artificial Intelligence, Bremen, Germany, September 15-17, 1998, Proceedings*, 81–92.

Edelkamp, S.; Kissmann, P.; and Torralba, Á. 2015. BDDs strike back (in AI planning). In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, 4320–4321.

Edelkamp, S. 2001. Planning with pattern databases. In *European Conference on Planning (ECP)*.

Edelkamp, S. 2005. External symbolic heuristic search with pattern databases. In *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005), June 5-10 2005, Monterey, California, USA*, 51–60.

Felner, A., and Ofek, N. 2007. Combining perimeter search and pattern database abstractions. In *Abstraction, Reformulation, and Approximation, 7th International Symposium, SARA 2007, Whistler, Canada, July 18-21, 2007, Proceedings*, 155–168.

Franco, S.; Torralba, Á.; Lelis, L. H. S.; and Barley, M. 2017. On creating complementary pattern databases. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 4302–4309.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1972. Correction to "a formal basis for the heuristic determination of minimum cost paths". *SIGART Newsletter* 37:28–29.

Helmert, M. 2008. *Understanding Planning Tasks: Domain Complexity and Heuristic Decomposition*, volume 4929 of *Lecture Notes in Computer Science*. Springer.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res.* 14:253–302.

Kvarnström, J., and Magnusson, M. 2003. TALplanner in the third international planning competition: Extensions and control rules. *J. Artif. Intell. Res.* 20:343–377.

Parberry, I. 2015. Solving the (n^2 - 1)-puzzle with 8/3 n^3 expected moves. *Algorithms* 8(3):459–465.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *J. Artif. Intell. Res.* 39:127–177.

Seipp, J., and Helmert, M. 2018. Counterexample-guided cartesian abstraction refinement for classical planning. *Journal of Artificial Intelligence Research* (62):535–577.

Seipp, J., and Helmert, M. 2019. Subset-saturated cost partitioning for optimal classical planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Sievers, S.; Katz, M.; Sohrabi, S.; Samulowitz, H.; and Ferber, P. 2019. Deep learning for cost-optimal planning: Task-dependent planner selection. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*.

Slaney, J. K., and Thiébaux, S. 2001. Blocks world revisited. *Artif. Intell.* 125(1-2):119–153.

Torralba, Á.; Alcázar, V.; Kissmann, P.; and Edelkamp, S. 2017. Efficient symbolic search for cost-optimal planning. *Artif. Intell.* 242:52–79.

Wah, B. W., and Chen, Y. 2004. Subgoal partitioning and global search for solving temporal planning problems in mixed space. *International Journal on Artificial Intelligence Tools* 13(4):767–790.

Performance Robustness of AI Planners to Changes in Software Environment

Chris Fawcett
University of British Columbia
fawcettc@cs.ubc.ca

Mauro Vallati
University of Huddersfield
m.vallati@hud.ac.uk

Alfonso E. Gerevini
Università degli Studi di Brescia
alfonso.gerevini@unibs.it

Holger H. Hoos
University of Leiden
hh@liacs.nl

Abstract

Solver competitions have been used in many areas of AI to assess the current state of the art, and to guide future research and real-world applications. AI planning is no exception, and the International Planning Competition (IPC) has been frequently run for nearly two decades. Due to the organizational and computational burden involved with running these competitions, solvers are generally compared using a single homogeneous software environment for all competitors.

In this work, we use the competing planners and benchmark instance sets from the 2014 IPC Agile and Optimal tracks to investigate two questions. First, how is planner performance affected by the specific choice of software environment? Second, is it a good strategy to run planners with more recent versions of their software dependencies, in order to maximise performance? By running these competition tracks on eight distinct software environments, we show that planner performance varies significantly based on the chosen software environment, that the magnitude of this variation differs considerably between planners, and that using more recent software versions is not always beneficial.

Introduction

Automated Planning has been studied extensively for several decades, resulting in automated planners being deployed in a variety of real-world applications. Part of the considerable progress in developing powerful domain-independent planners can be ascribed to the International Planning Competitions (IPCs). Competitions play an important role in many areas of AI, helping to drive forward research and development of solvers for prominent problems, as well as incentivising the development and distribution of related tools and benchmarks. Competitions also play a prominent role in assessing and improving the state of the art in solving challenging AI problems, such as planning.

While all of the planning systems participating in the IPC are available to be used after the competition, top-ranked planners receive much of the attention and often have considerable long-term impact on research as well as on real-world applications. For this to make sense, we need to assume that, at least from a qualitative point of view, conclusions derived from competition results generalise well to other – even significantly different – hardware and software environments than those used for running the competition. It is well-known that competition results are already strongly affected by the set of benchmark instances and the evaluation function used to assess planner performance, as well as by the way in which benchmarks are described, and by the set of competitors (Howe and Dahlman 2002; Long and Fox 2003; Hoffmann and Edelkamp 2005; Gerevini et al. 2009; Linares López, Celorrio, and Olaya 2015a; Vallati and Vaquero 2015). It also comes as no surprise that results are strongly affected by resource bounds – in particular, the running time cutoff and the amount of RAM available to the planner. Moreover, an analysis performed on the SAT competition showed that ranks of solvers are also affected by pseudo-random number seeds used in runs of randomised solvers (Hurley and O'Sullivan 2015).

Interestingly, a previous investigation performed by Howe and Dahlman (2002) showed that the relative (qualitative) performance of planners can vary, among other factors, also when run using different hardware environment configurations. However, as their work focused on identifying potential sources of performance variation, their analysis concentrated on assessing differences between two different machines having the same software environment configuration.

In this work, we present an investigation into the impact of software environment choices on competition outcomes. Our experimental analysis involves eight different software configurations, including the choice of C/C++ compiler version, Python interpreter version and Java version. We attempt to identify aspects that have unequal impact on planner performance, in order to emphasise those aspects that have to be carefully considered when interpreting competition results.

When dealing with such software configuration choices, a question naturally arises:

Is it better to use the latest available versions of software environment components such as compilers or interpreters, or stick to the specific environment used during the IPC?

To help address questions such as this, we investigate the performance variation of planners that took part in two deterministic tracks of the 2014 International Planning Competition: the *Optimal* and *Agile* tracks. The *Optimal* track is one of the longest-standing tracks in the IPC series, with many participating planners and substantial impact on the field of AI planning. While the *Agile* track was new for IPC 2014, its emphasis on planner running time and low resource requirements made it ideally suited for our analysis.

Our results show that competition rankings are strongly

affected by the software environment. For a more extensive analysis, which shows also the impact of different hardware configurations on planners, the interested reader is referred to (Bocchese et al. 2018).

Sources of Performance Variation

In this work, given the lack of analysis in the literature, we focus on the impact of software environment on planners' performance. However, there are many possible sources of performance variation that can affect empirical performance analyses and competition outcomes. Here, we briefly survey some of the most important of these sources of variation.

Planner randomization. Many planners take advantage of randomization to improve average-case performance and to avoid manual deterministic development choices. This randomization can result in very different planner trajectories in repeated runs with different random seeds, with a correspondingly wide variation in the resulting performance (Hurley and O'Sullivan 2015).

Running time and memory. Generally, increasing the running time or memory allocated to a planning system will result in more problem instances solved. However, this performance improvement will not be uniform across planners; for example, planners that perform extensive precomputation or use pattern databases tend to benefit more from increased memory limits (Linares López, Celorrio, and Olaya 2015b).

Hardware environment. It is clear that hardware choices, such as CPU type and speed, can affect planner performance, and it is known that planners are affected to varying degree by such differences in hardware environment (Howe and Dahlman 2002). Other aspects of the hardware environment that can have significant impact on planner performance include CPU cache, memory bandwidth, local storage medium, general-purpose graphics processing units (GPGPU) and network fabric.

Software environment. There are many aspects of the software environment configuration that can affect planner performance. These choices include the operating system and version, versions and compilation of system libraries (e.g., LIBC), as well as the version, linking, and compiler used for building all further software components required by a given planning system.

Benchmark instances. The instances used for evaluating planning systems should be challenging and need to allow for performance differences between planners to be identified. While it is clear that the use of different benchmarks can lead to very different results, we note that planning instances are often created using randomised generators, where a few parameters define the size and the difficulty of the resulting instances. The choice of problem instance domains, generator settings, as well as instance set size and distribution will all have an effect on planner performance (Howe and Dahlman 2002). Furthermore, also the order in which elements are listed in benchmarks has strong influence on planner performance (Vallati et al. 2015b; Vallati and Serina 2018).

Ranking mechanism. The metric used to assess planner performance (running time, instance set coverage, solution quality) and the techniques for aggregating performance across a given set of benchmark instances affect the outcome of empirical performance evaluations and competitions. Some competitions use an absolute scoring mechanism (such as mean running time), while others (such as earlier editions of the IPC) use relative scoring mechanisms, where the performance score of a planner is potentially affected by the performance of its competitors. The performance of a given planning system in relation to others can vary considerably depending on the ranking mechanism used (Linares López, Celorrio, and Olaya 2015b; Vallati, Chrpa, and McCluskey 2018).

	GCC	Python	JVM
gpj	4.7.2	2.7.3	1.7
Gpj	4.8.2	2.7.3	1.7
gPj	4.7.2	2.7.10	1.7
GPj	4.8.2	2.7.10	1.7
gpJ	4.7.2	2.7.3	1.8
GpJ	4.8.2	2.7.3	1.8
gPJ	4.7.2	2.7.10	1.8
GPJ	4.8.2	2.7.10	1.8

Table 1: The 8 software configurations considered in this investigation. Lowercase and uppercase are used to distinguish the "base" and "newer" configurations of each component.

Methodology

For our experimental analysis of the impact of the software environment on planner performance, we chose two sequential, deterministic tracks of the 2014 International Planning Competition (IPC): the *Agile* (15 participants) and *Optimal* (17 participants) tracks. These two tracks provide a very interesting test-bed, as they rank competitors using nearly opposite metrics. The competing planners are therefore likely to exploit significantly different approaches and techniques. In the *Optimal* track, planner running time is of limited importance: planners are assessed according to their ability to generate optimal solution plans within a given (large) cutoff time. In the *Agile* track, on the other hand, the quality of solutions is irrelevant, as planners are ranked according to their ability to quickly find a solution.

We chose to investigate three major software components in our analysis: GCC compiler version, Python interpreter version, and Java version. Nearly every planner which took part in IPC 2014 was entirely or partially reliant on components compiled with GCC, and different compiler versions are very likely to produce different executables even when identical command-line options are used. We selected GCC versions 4.7.2 and 4.8.2 as the two configuration options, since 4.7.2 was that used in the competition and several of the planners did not successfully compile with GCC versions more recent than 4.8.2.

Python and Java were by far the next most common software dependencies for the planners we considered. We selected Python 2.7.3 and Oracle Java 1.7.0_45, the versions used in IPC-2014, as well as Python 2.7.10 and Oracle Java 1.8.0_65, the most recent versions at the time of our experiments under which planners would run successfully.

The combination of these choices resulted in 8 poten-

tial software configurations, all of which were used in this work. Table 1 details each of the 8 configurations considered. For conciseness, each configuration will be referred to using one letter for each component: *g* for the GCC version, *p* for the Python version, and *j* for the Java virtual machine version. Since we consider two versions of each component, we use lowercase for indicating the *base* version, and uppercase when referring to the more recent version of the given component. Therefore, we denote the default configuration provided by the organisers of IPC 2014 as *gpj* (the *base* configuration), and the configuration with the more recent of each option as *GPJ* (the *newer* configuration).

Many of the planners that took part in the tracks we considered did require some modification in order to run successfully on our hardware and software configurations, for example to avoid writing temporary files into their source directories and polluting results when executing runs concurrently. We consider these modifications minor and do not believe that they had any effect on planner execution or running times. There were two exceptions, namely the *Freelunch* planner from the *Agile* track and the *AllPaca* planner from the *Optimal* track. In the case of *Freelunch*, we could not successfully run the planner on our computer cluster with any version of Java. As far as we can determine, this was caused by the high-memory shared environment on each cluster node, as *Freelunch* would crash immediately on launch with a Java JVM memory allocation exception. In the case of *AllPaca*, the planner relied on the presence of a specific commercial Lisp variant, and we were unable to modify it to work with any of the Lisp distributions available on our systems. These two planners have therefore been removed from our results, but we fully expect that if these issues were to be fixed, they would not significantly impact our results.

All the experiments were run on a cluster of homogeneous machines running CentOS version 5.0, each containing two Intel Xeon X5650 2.66GHz six-core processors with 12MB cache, and 24GB of available RAM. All planner runs were performed independently in parallel, with each run assigned one CPU core, 8GB of RAM, and the running time limits used in each track of IPC 2014. Running time and memory limits were monitored and enforced using tools from AClib (Hutter et al. 2014).

It is common practice in automated planning to include randomised components in planning systems. Randomisation is useful, e.g., for breaking ties during the heuristic search, introducing some noise in the heuristic search state evaluation, or performing search restarts. Evidently, planner performance can be affected by this source of stochasticity. In order to account for this and attempt to isolate the impact of software configuration on planner performance, our results for any given planner were obtained by averaging over five independent runs on each benchmark instance.

Empirical Analysis

Table 2 illustrates how the instance set coverage of the *Optimal* track planners are affected by our software configurations. Several planners exhibit a sizeable performance drop when Java 1.8 is used instead of version 1.7. The planners most affected by this are *MIPlan* and *NuCeLaR*; the planners based on *Gamer*, i.e., *Gamer*, *cGamer-bd* and *Dynamic-Gamer* also show a performance drop, although not as significant as for the previously-mentioned planners. We note that the performance variations observed for *MIPlan*, *NuCeLaR*, *Dynamic-Gamer* and *Gamer* were sufficient to cause changes in competition ranking between software environment configurations. The remaining planners of the *Optimal* track show minor performance fluctuation. Figure 1(a) provides a visual representation of the performance variation on each of the 8 considered configurations.

	Instance coverage for each software configuration							
Planner	gpj	Gpj	gPj	GPj	gpJ	GpJ	gPJ	GPJ
cGamer-bd	130.2	**131.4**	**131.4**	130.8	128.2	127.4	127	128.8
RIDA	117.2	115.8	**117.4**	116	**117.4**	116.2	117	116.4
Metis	112.2	112.4	112.6	112.6	**112.8**	112.6	112.4	111.8
SymBA-1	108.2	108.6	108.4	108.4	**108.8**	108.6	108.4	108.6
SymBA-2	108	108.8	108.2	108.6	108.6	**109**	108.4	108.4
MIPlan	112.6	112.6	**113.2**	113	102.2	103.2	103.4	103
D-Gamer	107.6	**110**	109	106.6	104.2	105.6	104.6	105.4
NuCeLaR	**108.8**	108.6	108.2	108.6	98.6	98.6	98.4	98.8
DPMPlan	101.4	101.8	102.2	102.2	102.4	**102.8**	102.4	102.4
Cedalion	95.4	96	96	**96.2**	95.2	95.8	95.6	95.8
Gamer	**96.8**	96.4	96.4	96.4	93.2	92.6	91.8	93.8
Rlazya	91	91.8	92.2	91.8	91.8	92	**92.6**	91.8
SPMaS	75.8	76.8	74.6	75.4	**77.2**	75.6	74.8	75.4
Hflow	56	57	56.8	**57.2**	56	**57.2**	56.6	57
Hpp	15	15	15	15	15	15	15	15
Hpp-ce	15	15	15	15	15	15	15	15

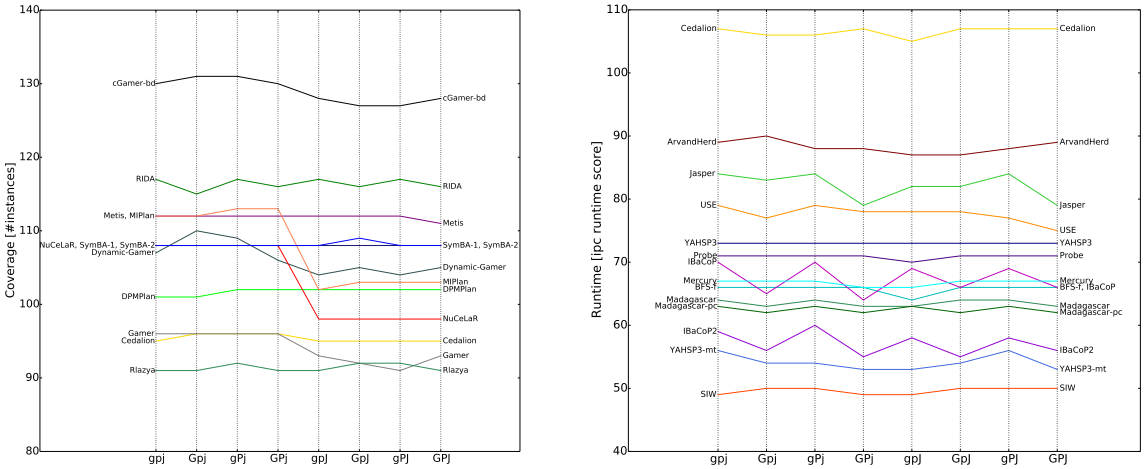
Table 2: Number of problem instances solved (instance coverage) for each of our 8 software configurations, using the IPC 2014 *Optimal* track planners and benchmark instance set. We present the mean coverage over 5 independent runs. Boldface is used to indicate the best performance achieved by a planner, in presence of variations.

	IPC score for each software configuration							
Planner	gpj	Gpj	gPj	GPj	gpJ	GpJ	gPJ	GPJ
Cedalion	107.4	106.2	106.9	107.6	105.8	107.1	**107.6**	107.0
IBaCoP	**70.1**	65.8	70.0	64.9	69.1	66.0	69.8	66.2
USE	79.3	77.7	**79.6**	78.1	78.1	78.1	77.6	75.7
ArvandHerd	89.2	**90.2**	88.1	88.2	87.9	87.5	88.8	89.8
IBaCoP2	59.6	56.6	**60.0**	55.7	58.1	55.6	58.6	56.1
Jasper	**84.8**	83.7	84.3	79.9	82.3	82.3	84.5	79.7
Mercury	**67.8**	67.5	67.8	66.5	66.4	67.7	67.1	67.0
BFS-f	66.1	66.4	66.2	**66.7**	64.7	66.1	66.1	66.5
Probe	71.4	71.2	71.3	71.4	70.7	71.4	**71.5**	71.2
YAHSP3	73.7	73.6	73.5	73.2	73.3	73.1	**73.8**	73.5
Madagascar-pc	**63.4**	62.6	**63.4**	62.6	63.1	62.6	63.2	62.4
YAHSP3-mt	**56.0**	54.2	54.2	53.8	53.7	54.4	**56.0**	53.4
Madagascar	**64.1**	63.9	64.0	63.9	63.9	**64.1**	64.0	63.3
SIW	49.6	50.2	50.1	49.8	49.4	**50.3**	50.0	50.1

Table 3: IPC score for each of our 8 software configurations, using the IPC 2014 *Agile* track planners and benchmark instance set. We present the mean coverage over 5 independent runs. Where performance variations exist for a given planner, we show the best performance achieved by that planner in boldface.

Table 3 shows how the software configuration affects planner performance for the *Agile* track. Performance is measured in terms of IPC score [1], which is focused on the running time required to find any satisficing plan; the quality of those plans is not considered. Some planners from this track demonstrated high sensitivity to the GCC com-

[1] https://helios.hud.ac.uk/scommv/IPC-14/rules.html



(a) IPC 2014 Optimal track

(b) IPC 2014 Agile track

Figure 1: Mean number of instances solved (instance coverage) and IPC score, respectively, using the planners and benchmark instance sets from the IPC 2014 *Optimal* 1(a) and *Agile* 1(b) tracks. We present results for each of the 8 software configurations.

piler version. For example, extreme variation can be observed in the performance of *IBaCoP* and *IBaCoP2*, to the extent of causing changes in competition ranking for *IBaCoP*. The performance of other *Agile* track planners, in particular *Use* and *Jasper*, are affected by a combination of GCC and Python versions. The other planners exhibit only minor performance variation.

Figures 1(a) and 1(b) show how the competition ranks are affected by the eight considered software configurations. It is easy to observe that the ranks are significantly influenced and many of the planners face changes in rank. Moreover, these results provide a valuable example of the impact of other sources of variation on the reproducibility of competition results: almost all of the considered planners were ranked differently in IPC 2014 official results.

Intuitively, one would expect the competing planners to achieve their best performance either on the *base* configuration (*gpj*) or on the *newer* configuration (*GPJ*). With regard to the former, the underlying hypothesis is that the planners' code should have been somehow "tuned" – for the sake of competition performance – for the configuration used in IPC 2014; this appears plausible, since participants had access to the competition cluster for testing their planners (Vallati et al. 2015a). On the other hand, it is also likely that optimisations introduced in newer versions of compilers or interpreters will be reflected in noticeable performance improvements of planning systems making use of them. Interestingly, for planners that participated in the *Agile* track, we observed a tendency for the *base* configuration to lead to the best performance (but still for only 6 of 14 competitors), while none of the planners achieved their best performance using the *newer* configuration. The situation is different for the *Optimal* track planners; Table 2 indicates that, while no planner achieves its best performance on the *newer* configuration, the *base* configuration is rarely the one providing the best results. Each planner is affected in a different way by the considered 8 software configurations.

Conclusion

Our analysis shows that the software environment in which a planner is run can substantially impact performance, and that this effect varies significantly among planners. These software environment changes can be as minor as the version of the compiler used to create each planner executable. Our analysis also indicates that running planners on software configurations that are more recent than those used in the competition can have surprisingly detrimental effects on performance.

While our experimental observations do suggest that competition performance results should be carefully interpreted, we caution that these observations should not be misunderstood as invalidating or diminishing the utility of planner competitions in general. Attempting to compensate for many of the sources of performance variation discussed in this paper would place a heavy burden on competition organisers, both in terms of time and additional computational resources. Allowing competitors the ability to customize their own software configuration for the competition, as done in the 2018 IPC using containers, could potentially reduce this source of variation, but could also have the side effect of biasing the competition results in favour of competitors with the expert knowledge, computational resources and time to finely tune their systems.

We see several possible avenues for future work: first, using the knowledge gained in this work, the study and development of a competition measuring solver performance across several distinct hardware and software environments; second, a thorough analysis of additional sources of performance variation not covered in (Bocchese et al. 2018), including benchmark instance set selection and solver stochasticity.

References

Bocchese, A. F.; Fawcett, C.; Vallati, M.; Gerevini, A. E.; and Hoos, H. H. 2018. Performance robustness of AI planners in the 2014 international planning competition. *AI Commun.* 31(6):445–463.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artif. Intell.* 173(5-6):619–668.

Hoffmann, J., and Edelkamp, S. 2005. The deterministic part of IPC-4: an overview. *J. Artif. Intell. Res. (JAIR)* 24:519–579.

Howe, A. E., and Dahlman, E. 2002. A critical assessment of benchmark comparison in planning. *J. Artif. Intell. Res. (JAIR)* 17:1–3.

Hurley, B., and O'Sullivan, B. 2015. Statistical regimes and runtime prediction. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, 318–324.

Hutter, F.; López-Ibáñez, M.; Fawcett, C.; Lindauer, M.; Hoos, H. H.; Leyton-Brown, K.; and Stützle, T. 2014. AClib: A benchmark library for algorithm configuration. In *Proceedings of the Eighth International Conference on Learning and Intelligent Optimization (LION'14)*, 36–40. Springer.

Linares López, C.; Celorrio, S. J.; and Olaya, A. G. 2015a. The deterministic part of the seventh international planning competition. *Artif. Intell.* 223:82–119.

Linares López, C.; Celorrio, S. J.; and Olaya, A. G. 2015b. The deterministic part of the seventh international planning competition. *Artif. Intell.* 223:82–119.

Long, D., and Fox, M. 2003. The 3rd international planning competition: Results and analysis. *J. Artif. Intell. Res.(JAIR)* 20:1–59.

Vallati, M., and Serina, I. 2018. A general approach for configuring PDDL problem models. In *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS*, 431–436.

Vallati, M., and Vaquero, T. 2015. Towards a protocol for benchmark selection in ipc. In *The 4th Workshop of the International Planning Competition*.

Vallati, M.; Chrpa, L.; Grzes, M.; McCluskey, T. L.; Roberts, M.; and Sanner, S. 2015a. The 2014 international planning competition: Progress and trends. *AI Magazine* 36(3):90–98.

Vallati, M.; Hutter, F.; Chrpa, L.; and McCluskey, T. L. 2015b. On the effective configuration of planning domain models. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, 1704–1711.

Vallati, M.; Chrpa, L.; and McCluskey, T. L. 2018. What you always wanted to know about the deterministic part of the international planning competition (IPC) 2014 (but were too afraid to ask). *Knowledge Eng. Review* 33:e3.

An Analysis of the Probabilistic Track of the IPC 2018

Florian Geißer
Australian National University, Australia
florian.geisser@anu.edu.au

David Speck
University of Freiburg, Germany
speckd@informatik.uni-freiburg.de

Thomas Keller
University of Basel, Switzerland
tho.keller@unibas.ch

Abstract

The International Planning Competition 2018 consisted of several tracks on classical, temporal and probabilistic planning. In this paper, we focus on the discrete MDP track of the probabilistic portion of the competition.

We discuss the changes to the input language RDDL, which give rise to new challenges for planning systems, and analyze each of the eight competition domains separately and highlight unique properties. We demonstrate flaws of the used evaluation criterion, the IPC score, and discuss the need for optimal upper bounds. An evaluation of the three top-performers, including their post-competition versions, and a brief analysis of their performance highlights the strengths and weaknesses of the individual approaches.

Introduction

Since 1998, the International Planning Competition (IPC) empirically evaluates state-of-the-art planning systems on various benchmark problems to promote research and highlight key challenges in AI planning. Initially, the competition focused on classical planning, but other, less restrictive competition tracks have emerged in subsequent years. Various tracks for reasoning under uncertainty have been added in 2004, and the probabilistic planning track, which is the focus of this paper, was organized for the sixth time in 2018.

Many different planning techniques have been applied by the participants over the years, ranging from determinisation-based re-planning (Yoon, Fern, and Givan 2007) over reinforcement learning with policy gradient methods (Buffet and Aberdeen 2007) and incremental policy refinement (Teichteil-Königsbuch, Kuter, and Infantes 2010) to Monte-Carlo tree search (Keller and Eyerich 2012; Keller and Helmert 2013). Not only planning techniques, but also benchmark problems have evolved over time: e.g., the dominance of determinisation-based approaches was answered with a shift to probabilistically interesting problems (Little and Thiébaux 2007), or the probabilistic PDDL dialect PPDDL (Younes and Littman 2004) was replaced by RDDL (Sanner 2010).

The latest probabilistic track also introduced a few changes. Providing challenging problems not only for the competition but also for future years was one of the announced aims of the organizers, so twice as many instances were created for each domain, scaling up to larger state and action spaces, and the used subset of RDDL was extended to support action preconditions, finite-domain, and intermediate variables to be able to model more realistic planning tasks. Additionally, time and memory limits were increased to allow offline approaches that compute an executable policy to compete with the predominant online approaches that interleave planning for a single state and execution of the last decision.

Five different planning systems participated in the competition. As up to two versions of each planning system could be submitted, seven planner variants were evaluated in terms of the IPC score in conjunction with the two PROST (Keller and Eyerich 2012) versions that won IPC 2011 and 2014 as a baseline. Although each participant had a unique approach, no planner dominated in every domain and the final results are very close – the winner, PROST-DD (Geißer and Speck 2018), and the two runner-ups, SOGBOFA (Cui and Khardon 2018) and Random Bandit (Fern, Issakkimuthu, and Tadepalli 2018) are separated by roughly five points.

In the first part of this paper, we describe the changes of IPC 2018 and analyse whether the competition lives up to its claim and introduces benchmarks with the potential of providing challenges for MDP research in upcoming years. We analyse the competition domains and provide properties that set them apart from other IPC domains. The second part of our work focuses on the evaluation metric that was used to determine the winner of the competition. We demonstrate that the IPC score is flawed if optimal state values of a problem are unknown, and we hope to spark a discussion among the planning community on this topic. The final part of the paper analyses the performance of the planners on the IPC 2018 benchmarks. We compare the results of bugfixed versions of the competition's top-performers and give insight into their strengths and weaknesses.

Input Language of the IPC 2018

As in the competitions in 2011 and 2014, the domains and instances of IPC 2018 were modeled in RDDL (Sanner 2010). In RDDL, both states and actions are described compactly by disjunct sets of parameterized variables and MDPs are specified as a dynamic Bayesian net with intermediate layers. There were some minor tweaks to the input language that are of no interest to this paper. The major changes on action preconditions and the introduction of finite-domain and

intermediate variables have an impact on the competition results and are discussed in the following.

Action preconditions. All RDDL domains from previous competitions come with a `state-action-constraints` section that contains a finite set of formulas P in first-order logic over the set of state- and action variables. Formulas containing at least one action variable form a constraint on the set of applicable actions a in a state s: if $s, a \not\models p$ for at least one $p \in P$, a is not applicable in s. Formulas without action variables, on the other hand, were introduced to provide invariants to a planning system and could be ignored by a planner. As the semantics of state-action constraints were never formally specified, it was unclear if an action is inapplicable in a state if its application can lead to a state where an invariant is violated. This cannot be checked efficiently by a planner as the number of outcomes can be prohibitively large. In previous competitions, this was irrelevant as all state-action constraints were static. For IPC 2018, state-action constraints were hence replaced by an `action-preconditions` section that contains formulas that have to be considered by each planner and checked on the current state before an action is applied, and an `invariants` section that may be ignored.

In previous competitions, the number $\overline{A} \in \mathbb{N}$ provided in the `max-nondef-actions` specification of RDDL instances has also been used to describe the applicability of actions. As it is possible to translate this for a given set of action variables $\{a_1, \ldots, a_n\}$ to the action precondition $\sum_{i=1}^{n} a_i \leq \overline{A}$, the `max-nondef-actions` entry was removed for simplicity of notation. A side effect of dynamic action preconditions is that action variables carry significantly more parameters at IPC 2018 than in previous competitions to be able to "connect" action preconditions and transitions functions (conditional probability functions or CPFs in RDDL). In turn, this leads to a higher average number of ground action variables in the IPC 2018 instances and, due to the exponential relationship between action variables and actions, to a significantly higher number of actions if action grounding is performed naively. While the set of actions that is applicable in at least one state (i.e., the actions that need to be grounded) is significantly smaller in most instances, it is in general not tractable to compute this set exactly, as determining if a given action is legal in at least one reachable state is at least as hard as the bounded plan existence problem in classical planning. Previously, a small value for $\overline{A}$ has helped to keep the number of ground actions small even if the grounding procedure is simple. This safety net has been removed along with the `max-nondef-actions` section, and new techniques need to be developed for the challenge of grounding RDDL actions. To make grounding simpler, the IPC 2018 domains guarantee that an action a with true action variables A is inapplicable in all reachable states if there is another action a' with true action variables $A' \subseteq A$ that is inapplicable in all reachable states.

Intermediate variables. A RDDL concept that has not been considered at IPC 2011 and 2014 are intermediate variables, which are typically used to determine the outcome of multiple interdependent stochastic effects. To illustrate the concept, assume that there is a 50% chance that two variables v_1 and v_2 are both true in the next state, and both are false otherwise. Modelling this with CPFs $v_i' = \text{Bernoulli}(0.5)$ for $i \in \{1, 2\}$, where Bernoulli is a RDDL keyword representing a Bernoulli distribution, would result in a model where all four possible value combinations come up with a probability of 25%. This does not reflect the desired transition dynamics. An intermediate variable v with CPF $v = \text{Bernoulli}(0.5)$ and CPFs for v_1 and v_2 of the form $v_1' = v$ and $v_2' = v$ lead to the described model, though.

Some of the IPC 2018 domains are modelled with intermediate variables, but planners were allowed to choose between a domain version with or a compilation without this feature. The compilation is performed by replacing intermediate variables with state variables and adding artificial intermediate decision steps where only a dummy action `proceed-interm-level` can be applied. There are further details to this compilation, e.g., variables are introduced to remember which action was executed and to represent the current level, and the horizon is increased according to the levels of the intermediate variables. However, these are not relevant for this paper and hence omitted.

Finite-domain variables. Finite-domain variables can be modelled in RDDL as enum-valued variables, a feature that has not been used in previous competitions. IPC 2018 made this feature available to planners that support it, but also provided a compilation of finite-domain variables into binary variables. In the classical planning setting, such a compilation can be performed by replacing each finite-domain variable v with domain $\text{dom}(v) = \{x_1, \ldots, x_n\}$ with binary variables $v\text{-is-}x_i$ for $i \in \{1, \ldots, n\}$. In the probabilistic setting, it is possible to do the same replacement, but the blowup is significantly larger. To illustrate this, consider a finite-domain variable v of type `enum_type` which is defined over the values x_1, x_2 and x_3, and assume v takes each value with uniform probability (modelled with the RDDL keyword `Discrete`). Due to the implicit dependency between the three values, we have to make sure that exactly one value becomes true in the next state, and a translation to

$$v\text{-is-}x_i' = \text{Bernoulli}(0.\overline{3});$$

for $i \in \{1, 2, 3\}$ is hence not correct. Instead, we have to sample these values consecutively, each time conditioned on the variables already sampled. As state variables are sampled at the same time, intermediate variables of the form

$$\begin{aligned} v\text{-is-}x_1 &= \text{Bernoulli}(0.\overline{3}); \\ v\text{-is-}x_2 &= \neg v\text{-is-}x_1 \cdot \text{Bernoulli}(0.5); \\ v\text{-is-}x_3 &= \neg v\text{-is-}x_1 \cdot \neg v\text{-is-}x_2; \end{aligned}$$

are used in the compilation to model the consecutive sampling (in increasing index order) properly. In this form,

- $v\text{-is-}x_1$ becomes true with probability $\frac{1}{3}$
- the Bernoulli(0.5) case of $v\text{-is-}x_2$ becomes relevant in the $\frac{2}{3}$ of the cases when $v\text{-is-}x_1$ did not become true, resulting in a probability of $\frac{2}{3} \cdot \frac{1}{2} = \frac{1}{3}$ and

- v-is-x_3 becomes true if neither of the former became true and hence also with probability $\frac{1}{3}$.

If the term inside the Bernoulli statement cannot be simplified as much as here, these terms quickly grow very large.

Competition Domains

In the following, we briefly introduce the domains that were used at IPC 2018 and highlight properties that make the domains particularly challenging. We base the presented information on the following sources[1]: 1) the output of a modified version of the PROST parser, which was enhanced with expert knowledge in some domains; 2) a random walk planner that computes averages over all states that are encountered in 200 runs; 3) and from computations by hand. In general, each domain consists of 20 instances, where the instances increase in size (in terms of states and actions), although not monotonically. While this has also been the case at IPC 2014, the largest instances of IPC 2018 (except for PUSH YOUR LUCK) are several orders of magnitude larger than the smallest ones.

ACADEMIC ADVISING is the only domain of IPC 2018 that has appeared in a previous competition, and it is equivalent to its predecessor apart from some minor changes that became necessary due to altered competition rules. However, neither of the 20 instances has been used before. In ACADEMIC ADVISING, a student takes courses at a given cost, aiming to complete a predefined subset of courses. The probability to pass a course increases with the number of previously passed prerequisites.

Prior to IPC 2018, ACADEMIC ADVISING has already been the domain with the largest number of applicable actions. However, if we compare the largest instances, that number grew from 466 in 2014 to more than 10^{11} in 2018, and the median over the instances increased from 43 to 1862, which poses a real challenge to the competitors.

CHROMATIC DICE is an MDP variant of the popular dice game Yahtzee, where up to five dice are rolled up to three times and show both values and colors (determined by independent stochastic processes). After rolling, the planner has to select a category and receives a reward depending on the faces of the dice and the selected category. At the end of the interaction, the planner may receive various bonuses if it performed well in certain category sections.

CHROMATIC DICE is special because it has by far the largest reward formula among all competition domains, consisting of formulas over almost 10000 state variables (most occur multiple times, but almost all state variables are among them at least once) and 24 (different) action variables. For a near-optimal policy, the boni become very important, and the planner has to plan ahead exceptionally far.

[1]Zenodo link for complete dataset used in this paper: `http://doi.org/10.5281/zenodo.3235174`

COOPERATIVE RECON is a significantly altered variant of the IPC 2011 domain RECON. In the 2018 version, the planner controls one or more planetary rovers that examine objects of interest in order to take a picture of detected life.

In most instances, there are several rovers, and collaboration between them is a key challenge. Rovers carry different equipment and have to share tasks among them to succeed, and they are also able to support other rovers in their tasks for a higher probability of success. This makes COOPERATIVE RECON special because the mutual application of two action fluents can be more valuable than the sum of its parts.

EARTH OBSERVATION is based on the domain by Hertle et al. (2014) where the planner controls a satellite orbiting Earth that has to take pictures of the landscape below, taking into account the current weather forecast (the presence of clouds when a picture is taken leads to poorer image quality and hence results in a lower reward).

While the branching factor induced by actions is very low in this domain – there are only 4 actions to move the camera and take a picture – the branching factor due to uncertainty is immense. The weather can change the current cloud cover drastically, and the number of successor states of a given state-action pair is tremendous, comparable only with the SYSADMIN domain of IPC 2011.

MANUFACTURER is a domain where the agent manages a manufacturing company that buys goods to use them in the production of other goods. The domain is modular in the sense that more and more options become available the more challenging the instance is. In the smallest instances, the agent only buys, produces and sells goods. More complex instances allow the construction of additional factories, hiring staff to influence the price or contracting a manager who enables the execution of more efficient actions.

All modules have in common that the agent has to accept an immediate cost for an increased long-term reward. This is already true for the basic buy-produce-sell cycle, and the horizon until the investment pays off gets larger and larger with more challenging instances. Additionally, this is one of the domains with the largest number of applicable actions and relevant preconditions (more than 10^6 in both cases).

PUSH YOUR LUCK is a single-player game where the main challenge lies in determining the optimal moment to stop a repeated stochastic process. The player rolls one or more dice repeatedly until they select to cash-out, yielding a reward that corresponds to the product of all rolled values. However, if the player plays too risky and a number comes up a second time, all rolled values are reset.

The instances of PUSH YOUR LUCK are among the smallest of IPC 2018. However, better play is rewarded disproportionately due to the exponential growth of the reward in the case of success and the total reset in the fail case.

RED-FINNED BLUE-EYE are a species of fish that are endemic to seven artesian springs in the Edgbaston Reserve in

Central Queensland, Australia. The species is critically endangered due to competition by the invasive Gambusia. This domain is inspired from the work of Nicol et al. (2017). A planner has to make sure that red-finned blue-eye do not become extinct, either by removing or poisoning Gambusia or by translocating red-finned blue-eyes. The domain is probabilistically interesting as the springs get connected only in the rain season depending on the (probabilistically determined) amount of rain.

RED-FINNED BLUE-EYE is challenging because it has the largest median number of actions (2680) and action preconditions (almost 10000), and more than 10^6 actions in the hardest instances. The planning horizon of up to 120 and a median of 90 is also the largest, which is particularly relevant because extinction of red-finned blue-eye leads to a disproportionately high penalty.

WILDLIFE PRESERVE is inspired from work of Nguyen et al. (2013) and Fang, Stone, and Tambe (2015) on rangers that protect a wildlife reserve from poachers by sending rangers to specific areas. Poachers attack parts of the reserve depending on their preferences and an expectation where rangers will likely show up. This expectation is computed by exploiting the assumption typically taken in Stackelberg Security Games that the defenders' (i.e., rangers) mixed strategy is fully observed by the attacker and memorized by poachers for a predefined number of steps.

In each step, the planner obtains a reward for each area that has not been attacked undefended, and a penalty for each area that has. The challenge is to predict where poachers will attack with high probability and to lure poachers into attacking an area where they are caught. Determinisation-based policies are informative in instances where the memory of poachers is short, but the quality decreases quickly when poachers remember more of the rangers' decisions.

Participants

We briefly introduce the competition participants and the underlying techniques they use. More details can be found in the planner abstracts that can be found on the competition website[2].

PROST (Keller and Eyerich 2012) is the winner of the two previous IPCs in 2011 and 2014 and participated non-competitively to serve as a baseline. PROST 2011 and PROST 2014 differ mostly in the used search algorithm: the former is based on the popular UCT algorithm (Kocsis and Szepesvári 2006), while the latter applies the UCT* algorithm of Keller and Helmert (2013). Both baselines use an iterative deepening search (IDS) on the (most-likely) determinised task to initialize action-values of search nodes that are added to the search tree.

The versions that were used for IPC 2018 are not exactly the same planners that competed in the previous competitions: bugfixes provided over the last few years were incorporated and a simple linear time distribution of the remaining time was used for each step. Additionally, the parser was updated to exploit the guarantee on action preconditions described before: the implementation starts to check potential applicability of an action with actions where only one action variable is true, and it iteratively adds more variables until a precondition is violated independently from the state.

PROST-DD is based on PROST 2014 and applies the UCT* algorithm. The planner differs from the baseline in the underlying action-value initializer function, the recommendation function used to select the action applied in each step, and does not use the baseline parser implementation. Instead, the PROST-DD parser performs resolution- and backtracking-based search in a similar fashion to the DPLL algorithm (Davis, Logemann, and Loveland 1962). Additionally, the performance of the evaluation of action preconditions was improved. As recommendation function the planner uses the *most played arm* recommendation (Bubeck, Munos, and Stoltz 2009), which favors the action which was selected the most in the root node of the search (the baseline planners favor the action with the best expected outcome).

For the heuristic function, the planner symbolically represents a deterministic version of the planning task as Algebraic Decision Diagrams (Bahar et al. 1993). It computes step-wise estimates in a similar fashion to symbolic value iteration (Hoey et al. 1999) and symbolic backward search (Speck, Geißer, and Mattmüller 2018). If the symbolic computation is not able to compute the estimates for a number of steps that is equal to the problem horizon, the planner performs iterative deepening search instead. Two versions of PROST-DD participated in the competition, which differ in the determinisation that is used in the heuristic: outcomes with probability smaller than 0.5 are pruned in one version, and smaller than 0.1 in the other.

Random Bandit is built upon the PROST framework and is based on the ϵ-greedy algorithm for multi-armed bandit problems, which estimates state values by simulating the greedy action with probability $1 - \epsilon$ and a random action otherwise. The parameter ϵ is set to 0.5. This decision in the root node is followed by a random walk whose simulation depth is initially determined as the minimum of 7 and an estimate that is based on the time required for IDS on the most-likely determinised task.

Conformant SOGBOFA is based on the work by Cui and Khardon (2016) and Cui, Marinescu, and Khardon (2018). It symbolically represents the state value function of the current state as an abstract syntax tree and searches for the best action by calculating gradients on this symbolic representation by means of automatic differentiation. One property of this representation is that computations are performed on the lifted action representation, which allows the planner to deal with large action spaces. Therefore, the planner does not explicitly ground actions. Two different versions of the SOGBOFA system participated in the competition: SOGBOFA-F and SOGBOFA-B. They differ in the way fractional values in the rollout policy are treated.

[2] ipc2018-probabilistic.bitbucket.io

A2C-Plan While all previous planning systems perform planning online, A2C-Plan is an offline planner working in two phases: in the training phase, a deep neural network is trained, using an actor-critic algorithm (Konda and Tsitsiklis 1999) which combines learning of a policy network and updating its parameters via gradient updates. The planner is built upon the PROST framework and therefore uses the baseline parser implementation.

Imitation-Net is another offline planner based on neural networks. It follows a supervised learning approach which generates training data by following a greedy one-step policy based on random sampling. Based on this training data, a policy network (Issakkimuthu, Fern, and Tadepalli 2018) is trained using stochastic gradient descent, which is used to select the best action in each step. Similar to A2C-Plan, the planner is built upon the PROST framework.

On Evaluation Metrics

The evaluation of the planner performance was in principle the same as in the previous competitions: planners performed a sequence of interactions (runs) with rddlsim (Sanner 2010) that simulate the execution of the planner's policy, and the average cumulative reward over those runs is used to estimate planner performance. However, some details were changed in comparison to previous competitions: the number of runs was increased from 30 to 75 for higher statistical significance of the averages; the number of instances per domain was doubled from 10 to 20 for better scaling between small and large instances; the horizon was instance dependent to add an additional challenge with short or large horizons; memory was doubled from 2GB to 4GB to reflect modern hardware; and the average deliberation time per step was raised from approximately 1 second to 2.5 seconds, resulting in a total deliberation time between approx. 1 hour for the instances with the smallest horizon of 20 and more than 6 hours for the instances with the largest horizon of 120, an amount of time that allows offline planners to come up with a policy that is competitive with online planning. Furthermore, planners had to announce prior to each run if the round contributes to the evaluation to ensure that an optimal policy also maximizes the average cumulative reward over the executions of the policy (Keller and Geißer 2015).

IPC Score. The quality of each planner is measured in terms of the IPC score with an artificial minimum policy that is set to the maximum of a random policy and (if possible) a policy that only executes no-op actions. The IPC score is a normalized value between 0 and 1, where 0 is assigned to a planner that did not simulate 75 runs successfully or did not outperform the minimum policy, 1 is assigned to the best planner in the competition (if there is a planner with 75 valid runs that performed better than the min policy) and a value that is linearly interpolated between these extremes is assigned to each of the other planners. In the following, we argue that without having access to an optimal upper bound, i.e., the average cumulative reward of an optimal policy, a planner evaluation based on the IPC score is flawed. We want to emphasize that the aim of this discourse is not to question which planner is the winner of the competition, but to spark a discussion among the community on the underlying evaluation metric, and to motivate further research on upper bounds for the benchmark sets used in planning.

Planner	A	B	Min.	Opt.
Instance 1	5.00	1.00	0	100
Instance 2	1.00	2.00	0	2
IPC Score w/o Opt.	**1.50**	1.20	0	-
IPC Score w. Opt.	0.55	**1.01**	0	3

Table 1: A hypothetical competition result, where the IPC score is evaluated with and without consideration of optimal upper bounds.

In principle, the problem is that without an optimal upper bound the IPC score is not a stable metric, as the introduction of a new participant may change the results of the scores of all other participants if the new participant performs best in *any* instance. A consequence is that introducing a new participant may change the ranking between the current participants. Even worse: after introducing optimal upper bounds on the rewards the winner might change.

Table 1 shows an example where the normalization with optimal rewards changes the outcome of a hypothetical competition. Here, without considering optimal expected reward, the performance of planner A appears to be better than the performance of planner B. More precisely, A is five times better than B in instance 1, while B is only twice as good as A in instance 2. According to the IPC score, planner A is the winner of the competition. After introducing upper bounds on the average cumulative reward, it turns out that planner B performs optimally in the second instance and in the first instance both planners perform poorly in comparison to the optimal policy. Now, according to the IPC score, planner B is the winner of the competition. Note that the order can also change with the introduction of a new planner who performs better in instance 1 than planner A: submitting similar planner configurations to the competition might be detrimental for the planner performance. With optimal upper bounds, the order of two planners can never change when a new planner is introduced, since the IPC values of these two planners do not change.

In the absence of optimal expected reward values, one option is to rank the planners in order of performance on each instance, instead of accumulating the relative average rewards. Such a ranking system is closely related to the field of social choice theory. Each instance induces a ranking which can be interpreted as an independent preference relation. In other words, each instance represents a voter and each planner a possible candidate. There are several election methods for determining a winner in such a ranking system. Each fulfill different criteria, such as independence from irrelevant alternatives, independence of clone alternatives, or the majority criterion. Yet, it is a well-known game theoretical result that no voting system can fulfill even a small number

of particularly important criteria (Osborne and Rubinstein 1994). Therefore, before any voting system can be applied, it is important to define the criteria one wants to meet.

While such a ranking system ignores the relative performance of the planners, the discussion shows that even for arguably simpler measures the final evaluation depends on the underlying evaluation criteria one wants to meet and guaranteeing multiple criteria might even be impossible. We also did not address the probabilistic aspect of the problem, where we have to consider variance on the possible rewards. Designing a satisfactory system is not only out of the scope of this paper, but also requires the input of the IPC community and a theoretical analysis and discussion of this topic.

Analysis of Planner Performance

One should keep in mind that the aim of the planning competition is not necessarily to elect the best planning system, but to promote research and highlight current challenges in planning. Thus, we now take a closer look at the competition results and discuss the strengths and weaknesses of the different systems. While the IPC score might not be the best metric to determine the best planning system, it still gives some insight into the planner performance on individual domains. In the following we will put our focus on the online planning systems, as both offline planners performed worse than the online planners in almost[3] all domains (total IPC score of 28.6 for Imitation Net and 26.6 for A2C-Plan), and leave a study of these systems for future work.

We begin our analysis by having a look at the official competition results, depicted in Table 2. Since the baseline planners PROST 2011 and 2014 were only provided to compare the current participants with the winners of the previous competitions, the official winner was PROST-DD. The differences between PROST-DD, SOGBOFA and Random Bandit are minor, though. Furthermore, all submitted planners contained more or less severe bugs at the time of the competition, which affected their performance on some of the domains. Since the goal of our analysis is to shed light on the current state of the art in probabilistic planning and exhibit potential future work, we performed an additional evaluation within the same competition setup, but used an updated version of the planners if available. IPC scores with an updated version of PROST-DD and SOGBOFA (Cui, Keller, and Khardon 2019) are depicted in Table 3. Note that the scores are computed without considering A2C-Plan and Imitation-Net results.

It is apparent that the bugfixes for both planners were quite significant on some of the domains and both PROST-DD and SOGBOFA significantly outperform the baseline planners in the bugfixed versions. Furthermore, PROST, PROST-DD and SOGBOFA each dominate all other planners in at least one domain with respect to the IPC score. As a consequence, we further focus on a brief analysis of only these three planners, look into possible reasons for their performance, and highlight possible future research to provide additional insight.

[3]The only domain where Imitation-Net showed competitive performance was WILDLIFE PRESERVE.

SOGBOFA is not only the planner which differs architecturally and algorithmitically the most from the otherwise PROST-based planners, the most recent version also significantly outperforms its competitors. The most prominent feature of SOGBOFA is the support of large action spaces (no grounding process is involved), thus one reason for the strong performance might be that the planner is simply able to at least do *something* on the larger domains. To see this, we compare the number of instances where each planner completed all 75 runs, depicted in Table 4. Indeed, there are 17 instances where SOGBOFA was the only planner able to complete all 75 runs. Note that none of the other planners uniquely completed 75 runs on any instance (considering only a single configuration per planner). However, recall that the IPC score is only affected if the planner outperforms the min-policy. This was only the case in 5 out of the 17 instances (1 COOPERATIVE RECON, 3 RED-FINNED BLUE-EYE, 1 MANUFACTURER), which indicates that the planner did not only perform well because it was able to handle many instances where the other planners failed.

The advantage of SOGBOFA in instances with large action spaces is further highlighted by its performance in RED-FINNED BLUE-EYE and the second half of the COOPERATIVE RECON instances. Both domains have a large action space, and although the other planners are able to outperform the min-policy in these domains, SOGBOFA has a significantly higher average reward. The third domain where SOGBOFA shines is the MANUFACTURER domain, yet the applicable action space of this domain is quite low (only a couple of actions are applicable in each step). As this is a domain where the SOGBOFA algorithm outperforms the THTS-based algorithms independently from the size of the problem, we see this domain as a candidate for future research on both algorithms. We also conjecture that for these three domains the relative performance indicated by the IPC score would still hold, even if an upper bound on these problems would be provided.

The only domain where SOGBOFA performed significantly worse than every other competitor is PUSH YOUR LUCK. Interestingly, this is the domain with the overall smallest problem size, both in terms of applicable actions and state-space size. This allows us to compute the optimal reward for some of the instances and compare it to the planner results, depicted in Figure 1. While THTS-based planners often reach the optimal expected reward, this does not hold for SOGBOFA, which might be attributed to the optimality guarantees of UCT*, whereas the automatic differentiation algorithm does not provide such guarantees.

PROST-DD showed the best performance in ACADEMIC ADVISING and WILDLIFE PRESERVE. In the following we will focus on a comparison between the PROST-DD planner and the baseline, as both planners share the underlying algorithm but differ in initialization, recommendation function, and the grounding process. We will focus on the initialization and the grounding process, and refer to Keller (2015) for a comparison of the behaviour of the different recommendation functions.

	PROST		PROST-DD		Random	SOGBOFA
	2011	**2014**	**0.1**	**0.5**	**Bandit**	
ACADEMIC ADVISING (20)	3.2	3.3	5.8	**6.6**	0.7	4.1
CHROMATIC DICE (20)	12.8	10.1	7.6	7.5	17.1	**19.4**
COOPERATIVE RECON (20)	9.0	10.7	10.3	**12.0**	1.5	6.9
EARTH OBSERVATION (20)	18.7	**19.9**	6.5	5.3	12.8	7.4
MANUFACTURER (20)	**7.1**	2.7	3.3	2.8	4.1	0
PUSH YOUR LUCK (20)	6.3	14.2	**15.0**	12.7	13.1	1.4
RED-FINNED BLUE-EYE (20)	6.9	6.0	5.9	5.4	5.6	**18.3**
WILDLIFE PRESERVE (20)	3.9	7.9	**14.3**	**14.3**	10.8	4.8
Sum (160)	67.9	74.7	**68.8**	66.5	65.6	62.3

Table 2: Official IPC scores of the top performers of the International Planning Competition 2018.

	PROST		PROST-DD		Random	SOGBOFA
	2011	**2014**	**0.1**	**0.5**	**Bandit**	
ACADEMIC ADVISING (20)	4.06	3.33	**6.84**	5.61	0.81	4.86
CHROMATIC DICE (20)	12.91	10.04	9.82	10.49	16.97	**19.2**
COOPERATIVE RECON (20)	6.19	6.85	7.82	8.06	1.33	**15.11**
EARTH OBSERVATION (20)	18.76	**19.73**	17.24	16.77	12.74	11.52
MANUFACTURER (20)	4.05	2.08	3.08	4.72	1.14	**10.34**
PUSH YOUR LUCK (20)	6.57	14.61	14.99	**15.22**	13.22	2.32
RED-FINNED BLUE-EYE (20)	6.41	7.32	5.9	4.92	5.49	**18.97**
WILDLIFE PRESERVE (20)	3.99	7.98	**15.87**	14.99	10.78	8.68
Sum (160)	62.94	71.94	81.56	80.87	62.48	**91.0**

Table 3: IPC scores of the IPC 2018 top performers, based on updated planner versions.

	PROST		PROST-DD	Random	SOGBOFA
	2011	**2014**	**0.1**	**Bandit**	
ACADEMIC ADVISING (20)	11	12	14	11	**20**
CHROMATIC DICE (20)	**20**	**20**	**20**	**20**	**20**
COOPERATIVE RECON (20)	13	15	17	17	**20**
EARTH OBSERVATION (20)	**20**	**20**	**20**	**20**	**20**
MANUFACTURER (20)	10	11	11	11	16
PUSH YOUR LUCK (20)	9	17	**20**	**20**	**20**
RED-FINNED BLUE-EYE (20)	11	15	15	17	**20**
WILDLIFE PRESERVE (20)	4	9	16	16	10
Sum (160)	98	119	133	132	**146**

Table 4: Number of instances for each planner where all 75 runs were completed in time.

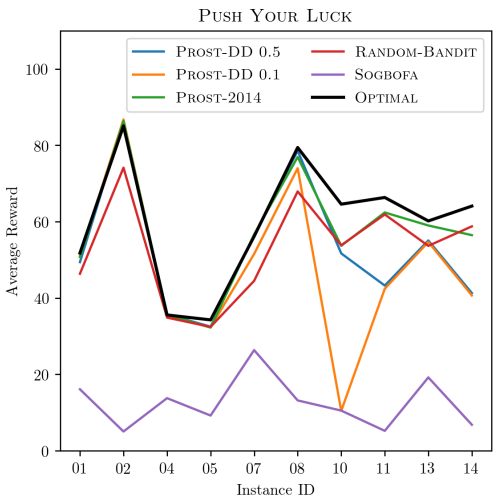


Figure 1: Average Reward in Push Your Luck.

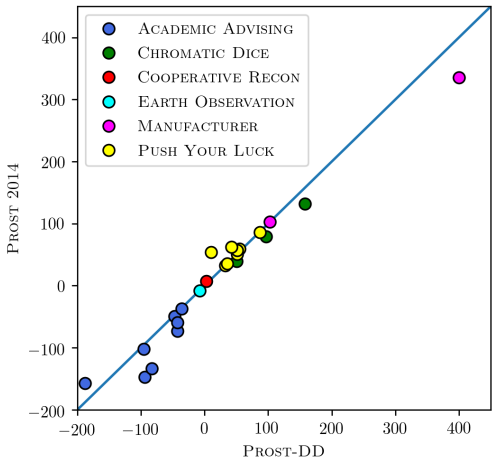


Figure 2: Average reward on instances where the DD heuristic was fully constructed.

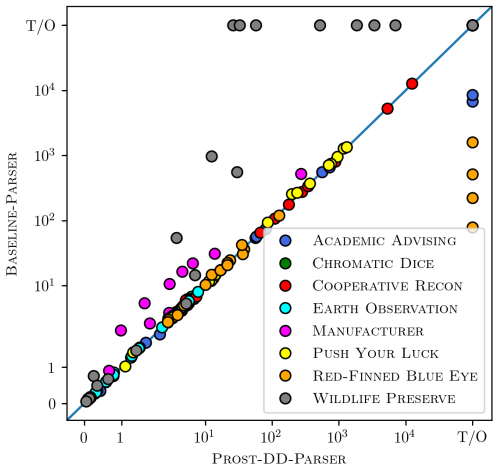


Figure 3: Parsing times in seconds.

As a first step, we evaluate the impact of the decision diagram based initialization. Note that in instances where the decision diagrams are not constructed, both planners rely on the same initialization (IDS). Figure 2 plots the average reward of Prost-DD and of Prost 2014 for each instance where both planners finished all 75 runs and where the decision diagram data structure was completely built up[4]. In general, the heuristic improves the average reward if constructed, which is also the reason for the stronger performance in Academic Advising. The varying performance in Push Your Luck is a result of both heuristics being very uninformative: in instances where a dice has more than 6 faces the probability for each face to appear becomes 0 in the determinisation. It is worth to note that for instances where the heuristic was not constructed, Prost-DD wastes up to 12.5% of the search time.

Next, we analyse the impact of the parser difference. Both parsers generate the same grounded instance, but differ in execution time. Figure 3 depicts the parsing time in seconds per instance. Clearly, the advantage of Prost-DD in Wildlife Preserve is due to the timeout of the baseline parser in half of the instances. On the other hand, in some instances of Red-Finned Blue-Eye and Academic Advising the Prost-DD parser times out while the baseline parser is able to parse the instance. Due to the poor performance of the baseline on these instances this does not influence the final score, though.

Prost. We conclude our brief planner analysis with a few words on the performance of the UCT* search algorithm, which is the core of both Prost 2014 and Prost-DD. The two domains where this approach significantly outperformed SOGBOFA are Earth Observation and Push Your Luck. For Push Your Luck, we have already seen that part of the reason is the optimality guarantee given by the Bellman backups of UCT*, which allows to compute the optimal expected reward for many of the instances (this also holds for Wildlife Preserve). It would certainly be interesting to see how close the Earth Observation results are to the optimal values. Additionally, Earth Observation is the domain with the least number of actions: only 4 actions are applicable in every state, which apparently favors sampling-based techniques.

Conclusion

To keep the conclusion brief, we emphasize the importance of having access to (near-) optimal rewards for the computation of IPC scores. Our analysis indicates that the benchmark set of IPC 2018 provides a challenge for the current state of the art in probabilistic planning and gives insight on possible future research. Future work on the Prost planner should be concerned with how to efficiently deal with large state and action spaces. For the SOGBOFA system, an interesting question is if it is possible to provide optimality guarantees. Why offline planners were unable to meet the performance of online systems remains an open question.

[4]We removed data points for Wildlife Preserve, as both planners share the average reward in 8 out of 9 instances.

Acknowledgments. Florian Geißer was supported by ARC project DP180103446, "On-line planning for constrained autonomous agents in an uncertain world". David Speck was supported by the German National Science Foundation (DFG) as part of the project EPSDAC (MA 7790/1-1). Thomas Keller received funding for this work from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 817639).

References

Bahar, R. I.; Frohm, E. A.; Gaona, C. M.; Hachtel, G. D.; Macii, E.; Pardo, A.; and Somenzi, F. 1993. Algebraic decision diagrams and their applications. In *Proc. ICCAD 1993*, 188–191.

Bubeck, S.; Munos, R.; and Stoltz, G. 2009. Pure exploration in multi-armed bandits problems. In *ALT*, volume 5809 of *LNCS*, 23–37.

Buffet, O., and Aberdeen, D. 2007. FF + FPG: Guiding a policy-gradient planner. In *Proc. ICAPS 2007*, 42–48.

Cui, H., and Khardon, R. 2016. Online symbolic gradient-based optimization for factored action MDPs. In *Proc. IJCAI 2016*, 3075–3081.

Cui, H., and Khardon, R. 2018. The SOGBOFA system in IPC 2018: Lifted BP for conformant approximation of stochastic planning. In *IPPC-6 planner abstracts*.

Cui, H.; Keller, T.; and Khardon, R. 2019. Stochastic planning with lifted symbolic trajectory optimization. In *Proc. ICAPS 2019*.

Cui, H.; Marinescu, R.; and Khardon, R. 2018. From stochastic planning to marginal MAP. In *Proc. NeurIPS 2018*, 3085–3095.

Davis, M.; Logemann, G.; and Loveland, D. W. 1962. A machine program for theorem-proving. *Communications of the ACM* 5:394–397.

Fang, F.; Stone, P.; and Tambe, M. 2015. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *Proc. IJCAI 2015*, 2589–2595.

Fern, A.; Issakkimuthu, M.; and Tadepalli, P. 2018. Random-bandit: An online planner. In *IPPC-6 planner abstracts*.

Geißer, F., and Speck, D. 2018. Prost-DD – utilizing symbolic classical planning in THTS. In *IPPC-6 planner abstracts*.

Hertle, A.; Dornhege, C.; Keller, T.; Mattmüller, R.; Ortlieb, M.; and Nebel, B. 2014. An experimental comparison of classical, FOND and probabilistic planning. In *Proc. KI 2014*, 297–308.

Hoey, J.; St-Aubin, R.; Hu, A.; and Boutilier, C. 1999. SPUDD: Stochastic planning using decision diagrams. In *Proc. UAI 1999*, 279–288.

Issakkimuthu, M.; Fern, A.; and Tadepalli, P. 2018. Training deep reactive policies for probabilistic planning problems. In *Proc. ICAPS 2018*, 422–430.

Keller, T., and Eyerich, P. 2012. PROST: Probabilistic planning based on UCT. In *Proc. ICAPS 2012*, 119–127.

Keller, T., and Geißer, F. 2015. Better be lucky than good: Exceeding expectations in MDP evaluation. In *Proc. AAAI 2015*, 3540–3547.

Keller, T., and Helmert, M. 2013. Trial-based heuristic tree search for finite horizon MDPs. In *Proc. ICAPS 2013*, 135–143.

Keller, T. 2015. *Anytime Optimal MDP Planning with Trial-based Heuristic Tree Search*. Ph.D. Dissertation, University of Freiburg.

Kocsis, L., and Szepesvári, C. 2006. Bandit based Monte-Carlo planning. In *Proc. ECML 2006*, 282–293.

Konda, V. R., and Tsitsiklis, J. N. 1999. Actor-critic algorithms. In *Proc. NIPS 1999*, 1008–1014.

Little, I., and Thiébaux, S. 2007. Probabilistic planning vs replanning. In *ICAPS 2007 Workshop on the International Planning Competition: Past, Present and Future*.

Nguyen, T. H.; Yang, R.; Azaria, A.; Kraus, S.; and Tambe, M. 2013. Analyzing the effectiveness of adversary modeling in security games. In *Proc. AAAI 2013*, 718–724.

Nicol, S.; Sabbadin, R.; Peyrard, N.; and Chadès, I. 2017. Finding the best management policy to eradicate invasive species from spatial ecological networks with simultaneous actions. *Journal of Applied Ecology* 54(6):1989–1999.

Osborne, M. J., and Rubinstein, A. 1994. *A course in game theory*. MIT press.

Sanner, S. 2010. Relational dynamic influence diagram language (RDDL): Language description.

Speck, D.; Geißer, F.; and Mattmüller, R. 2018. Symbolic planning with edge-valued multi-valued decision diagrams. In *Proc. ICAPS 2018*.

Teichteil-Königsbuch, F.; Kuter, U.; and Infantes, G. 2010. Incremental plan aggregation for generating policies in MDPs. In *Proc. AAMAS 2010*, 1231–1238.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. FF-Replan: A baseline for probabilistic planning. In *Proc. ICAPS 2007*, 352–360.

Younes, H. L. S., and Littman, M. L. 2004. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical Report CMU-CS-04-167, Carnegie Mellon University, School of Computer Science.

Benchmarks Old and New: How to compare domain independence for cost-optimal classical planning?

Ionut Moraru and Stefan Edelkamp
Informatics Department, Kings College London
Strand Campus, Bush House, 30 Aldwych, London, WC2B 4BG
ionut.moraru@kcl.ac.uk and stefan.edelkamp@kcl.ac.uk

Abstract

Domain independence is one of the main features of automated planning. Planners, in the context of cost-optimal classical planning, are developed with the intent of solving any type of problem that can be formulated in PDDL. We then compare planners by the number of problem instances they solve on a set of benchmarks, one point for each problem solved. However, does solving the most problems automatically result in having the best domain-independent planner?

In this paper, we compare the best performing, non-portfolio planners from the cost-optimal classical track of the International Planning Competition (IPC) 2018 on the complete set of benchmarks from the previous two competitions (2011 and 2014) and on a subset of the competitions from before 2011. Results show that, as the number of problems for each domain varies, current way of comparing planners (total coverage) can be biased towards the planners that perform the best in the domains with the most instances, but once we normalize those results, we can get a better picture for which technique is the most domain independent.

1 Introduction

Marvin Minsky classified Artificial Intelligence into five areas (1961), one of them being *planning*. Planning is the discipline that has the task of coming up with a sequence of actions that, starting from an initial state, will achieve a goal.

A technique that has appeared in the 80's and which has been giving really good results is the creation of solver software for resolving well-defined mathematical models, i.e. Constraint Satisfaction, Linear Programming, etc. (Geffner 2014). They are a general type of software, created for computing the solution of any problem specified in its input modeling language. In the field of AI Planning, this type of solvers are called *planners*, most of them using as an input PDDL (McDermott 1998; Fox and Long 2003).

Domain-independence is one of the qualities that planning as an area of research strives to achieve. What this means is that, as long as the problem is specified properly in PDDL, a planner should be able to give a valid plan as an output (Howey, Long, and Fox 2004). This is an extremely ambitious goal, especially when taking into consideration the complexity of a planning task (Bäckström and Nebel 1995; Bylander 1994). Nonetheless, current planners are able to solve a large variety of problems from very different domains (from solving the Rubik's cube and Solitaire games, to Logistics and path-finding problems just as an example) which cannot be described as anything but incredible.

The current way of comparing planners in the setting of domain-independent classical planning is by seeing how many problems each planner can solve. Each problem validly solved gives towards the planner one point. The planner with the most *points* can be considered the *best* on the tested benchmarks. While this way has its values, having a planner that solves the most problems is a feat that should be celebrated, in our view this does not capture the complete picture of domain-independence.

In this paper, we will be arguing that classical planners should be compared in more ways than just how many problems they can solve. We will be taking the four best performing, non-portfolio, cost-optimal planners from the 2018 International Planning Competition and compare them on a larger set of problems. We have seen that, because of the different number of problems for each domain, some domains are more *important* than others when comparing just the total coverage over the benchmark set. We argue for the introduction of a normalized domain coverage metric, which would alleviate this issue and would be more representative for comparing planners in domain-independent planning.

2 International Planning Competition

The International Planning Competition (IPC) has been a great driver for progress of research and has brought forth many novel techniques and planning technologies since it's inception in 1998. Organized together with the International Conference of Automated Planning and Scheduling, it has build an identity synonymous with state-of-the-art for planning in any of its forms (in 2018 we had Classical, Probabilistic and Temporal tracks).

At the beginning, events were held every two years, as the planning research in the modern sense was in developing fast, but recently, as the benchmark sets available were larger and more planners were broadly available for inspection, advances have slowed down. Competitions are now organized every 3-4 years, giving time for researchers to advance the field and implement any new idea.

Importance

IPC have brought a lot of benefits for the subject as a whole, first and foremost with PDDL, the high-level modeling lan-

guage that has now become an informal standard input for most planners. PDDL was used from the first edition, bringing all the new versions and new features for one of the subsequent editions. As all the domains and problems are formulated using this modeling language, almost all modern planners are built now to support one of the versions of PDDL and more recently RDDL for Probabilistic Planning (Sanner 2010).

Continuing on the topic of benchmarks, each edition published either completely new or reinterpretations of domains with new problems, increasing number and diversity of available benchmarks for the planning community. This gives planner developers a more complete way of evaluating their systems.

Finally, competitions in any field bring together any comunity and it manages to evolve a field. Comparing in a closed environment a vast number of planners, each approaching problems in a different way, has the benefit of putting head-to-head each method without bias. As the benchmarks are not know prior to the planner submission, developers of said systems need to focus on creating domain-independent planners, suited for any possible domain.

Planning Evolution

After each IPC, certain techniques have risen as the *state-of-the-art*. In the past, heuristic search was most of the time the best approach, and certain heuristics were highly successful (Helmert and Domshlak 2009). Symbolic search has also had success with SymBA*, a symbolic bidirectional planner (Torralba et al. 2014).

Each winner of the competition has shown the planning community which combination of technique and domain works especially well. Most the best performing planners have been awarded more attention in the following years, bringing forth their ideas in the community. Also, each well performing planner in the competition has made the organizer of the following competition to make their benchmark set harder for those techniques. This has made the community now to have a very diverse set of problems on which we can see how well each planner performs.

Portfolio planning is a technique that tries to combine find the best planner for the domain/problem that it has to solve. Work done by Sievers et al. (2019) has shown how grounded problems can be classified as to give the *better suited* planner the problem to solve. Following this work, portfolio planners are aiming to find the planners best suited for a type of planning task. This is a different approach to domain-independent planning, but has shown good results when looking at the results from the 2018 IPC.

3 Measuring Cost-Optimal Planning

In this section we will be discussing different ways of comparing planners, and see how they differ from each other. We have tested five planners, Complementary 1 and 2, Planning-PDBs, Scorpion and Symbolic-Bidirectional on a 69 domains, all the benchmarks from the previous three competitions and a subset of the domains from before 2011.

Coverage

As stated in the first section, the current way of comparing cost-optimal classical planning is by measuring the coverage of a planner (i.e. how many problems a planner can solve on a set of problems). Each problem solved is counted as a point towards that planner and at the end we compare the tally of each planner, the one with the most being the winner.

This metric is used both in competitions and in published papers when measuring the performance of a new method. However, this metric can become domain dependant if the number of problem instances is not uniform over all the domains. In our pre-2011 set of problems, made out of 31 domains, we can see that some domains are a lot more important than others when using this approach (seen in figure 1).

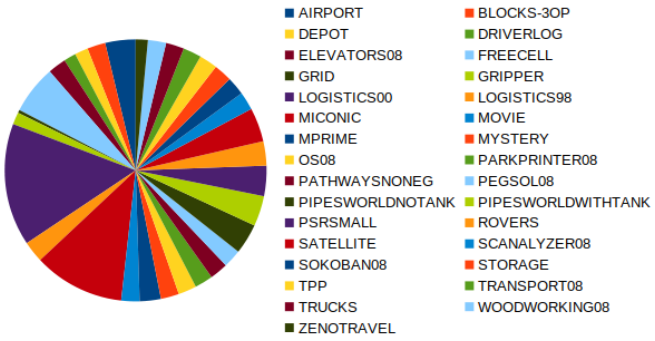


Figure 1: Size of domains from our pre-2011 benchmark

The benchmarks from 2011 and 2018, the domains were kept at a uniform size of 20 instances each. In that case, there is no need to normalize the results, but when using becnhmark sets like 2014 (most had 20 instances, with three different) and the pre-2011 we used (from 5 to 202, with most having 30 instances), the change in domain sizes requires a change in the evaluation metric.

	Problems Solved	Coverage	Normalized Coverage
Planning-PDBs	1122	54.17%	59.42%
Complementary1	1099	53.06%	57.60%
Complementary2	1164	56.15%	**62.08%**
Scorpion	**1208**	**58.32%**	60.11%
Sym-BiDir	1053	50.84%	55.46%

Table 1: Overall results as number of problems solved, coverage and normalized coverage.

Normalized Domain Coverage

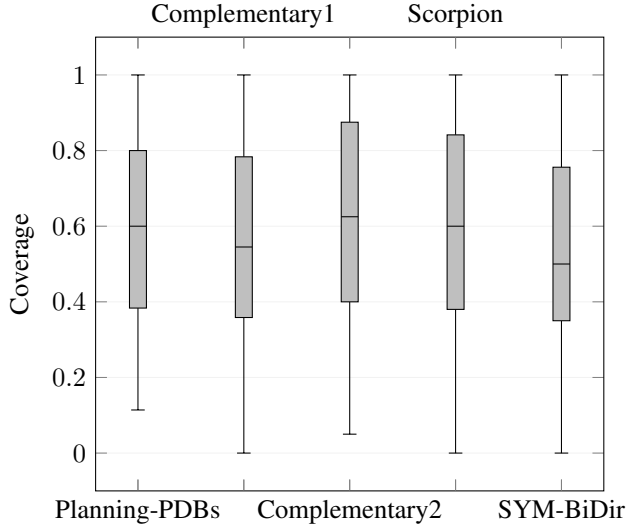
For cases like this, we normalize the domain coverage, and then get the average for each planner. By doing this, we first see how much of a domain a planner can solve, and then by averaging we get a better metric for overall domain-independent performance of a planner.

We can see the value of such a metric in table 1, where we can see that, even though Scorpion solves the most probles

	Pre 2011	Coverage	Normalized Coverage	IPC11	Coverage (also Normalized)	IPC14	Coverage	Normalized Coverage	IPC18	Coverage (also Normalized)
Planning-PDBs	678	50.78%	55.88%	190	67.85%	131	51.17%	53.48%	123	61.5%
Complementary1	680	50.93%	55.95%	185	66.07%	111	43.35%	46.15%	123	61.5%
Complementary2	686	51.38%	56.95%	**198**	**70.7%**	**155**	**60.54%**	**61.99%**	**124**	**62%**
Scorpion	**785**	**58.80%**	**60.20%**	190	67.85%	118	46.09%	48.77%	104	52%
Sym-BiDir	647	48.45%	53.46%	174	62.14%	129	50.39%	52.97%	114	57%

Table 2: Results of the five planners on the pre-2011, IPC11, IPC14 and IPC 18 benchmarks. For each benchmark we have the number of problems solved, coverage and normalized coverage (where needed).

out of the total of 2071 we tested on, the normalized coverage is worse than the one of Complementary2 (62.08% to 60.11%).



By looking at the boxplot of the normalized per-domain coverage of each planner, we can see an even clearer picture. The only two planners to solve problems on all the 69 domains are Planning-PDBs and Complementary2. Also, Planning-PDBs is a lot closer to Scorpion than what the number of instances solved would imply (1208 to 1122).

Another way of measuring the perfocmance when having a normalized coverage, would be by getting the median value for each planner. In our test cases, we find that Complementary2 has the best with 65%, with Scorpion and Planning-PDBs following (both have 60%). Complementary1 and Symbolic-Bidirectional finish the top with 55% and 50%.

4 Future suggestion

We can see that having the same number of problem instances for each domain is vital for evaluating domain independence. The organizers of IPC11 and IPC18 saw this and had an uniform number of problems. But in the future, by using a normalized domain coverage, future organizers can break from this constraint. Some domains would need a more granularity to differentiate the planners (domains where most planners get the same coverage). While organizing the competition and seeing this, organizers can add more instances for those domains.

Also, each IPC has contributed with new domains and problems that have been added by the community for evaluating their planners and subsequently adding them to their results sections in conference and journal publications. From just a glance at the problems we evaluated on, we can identify that from the current number of available domains there are more instances per domain from before IPC11, with an average of 30, and since 2011 having an average of 20. This will make in any evaluation section the domains from before 2011 of a more importance and not have a fair comparison of the domain-independence of a planner.

5 Conclusion

In this short paper, we propose that we should compare planners not only by the number of problems solve, but also by normalized pre-domain coverage as to evaluate the domain independence of a planner.

We do not want to subtract any value from the previous method. Solving more problems will always be a great indicator to the performance of the planner. However, due to the nature of our current set of problems, we identify that the variance in number of instances per domain is an issue and could lead to planner developers focusing their attention to solving the domains with the most instances.

We do not touch on any other metric of evaluating a certain technique in cost-optimal planning. There are many other ways that we can use to show an even more complete picture of a planner, but that is for future work.

References

[1995] Bäckström, C., and Nebel, B. 1995. Complexity results for sas+ planning. *Computational Intelligence* 11(4):625–655.

[1994] Bylander, T. 1994. The computational complexity of propositional strips planning. *Artificial Intelligence* 69(1-2):165–204.

[2003] Fox, M., and Long, D. 2003. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research* 20:61–124.

[2014] Geffner, H. 2014. Artificial intelligence: From programs to solvers. *AI Communications* 27(1):45–51.

[2009] Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: what's the difference anyway? In *ICAPS*, 162–169.

[2004] Howey, R.; Long, D.; and Fox, M. 2004. Val: Automatic plan validation, continuous effects and mixed initia-

tive planning using pddl. In *16th IEEE International Conference on Tools with Artificial Intelligence*, 294–301. IEEE.

[1998] McDermott, D. 1998. The 1998 ai planning systems competition. In *AI Magazine*, 35–55.

[1961] Minsky, M. 1961. Steps toward artificial intelligence. *Proceedings of the IRE* 49(1):8–30.

[2010] Sanner, S. 2010. Relational dynamic influence diagram language (rddl): Language description. *Unpublished ms. Australian National University* 32.

[2019] Sievers, S.; Katz, M.; Sohrabi, S.; Samulowitz, H.; and Ferber, P. 2019. Deep learning for cost-optimal planning: Task-dependent planner selection.

[2014] Torralba, A.; Alcázar, V.; Borrajo, D.; Kissmann, P.; and Edelkamp, S. 2014. Symba: A symbolic bidirectional a planner. In *International Planning Competition*, 105–108.

Planner Metrics Should Satisfy Independence of Irrelevant Alternatives (Position Paper)

Jendrik Seipp
University of Basel
Basel, Switzerland
jendrik.seipp@unibas.ch

Abstract

We argue that planner evaluation metrics should satisfy the independence of irrelevant alternatives criterion, i.e., the decision whether planner A is ranked higher or lower than planner B should be independent of planner C. We show that three metrics used in classical planning competitions do not necessarily satisfy this criterion and highlight alternative metrics that do so.

Introduction

Arrow's impossibility theorem is an important result from social choice theory (Arrow 1950). One of the fairness criteria it suggests is *independence of irrelevant alternatives* (IIA). In the setting of planning competitions, IIA translates to the requirement that the decision whether planner A is ranked higher or lower than planner B must depend only on the performance of planners A and B and not on another planner C. We believe that IIA is a critical requirement for planner evaluation metrics.

In the following, we show three planner evaluation metrics that do not satisfy IIA and give alternative metrics that do satisfy it.

IPC Satisficing Track

In the satisficing track of the International Planning Competition (IPC) planners are given 30 minutes to find plans. The time for finding plans is ignored, but cheaper plans are preferred. More precisely, the track uses the following metric (which we call *sat*) to evaluate a planner *P* on task π: *P* gets a score of 0 if it fails to solve π within the resource limits and a score of $\mathit{Cost}^*/\mathit{Cost}$ if it solves π, where *Cost* is the cost of the cheapest plan that *P* finds for π and Cost^* is the cost of a *reference* plan, i.e., a cheapest known plan for π. The total score for a planner is the sum of its scores over all tasks.

It is easy to see that *sat* satisfies IIA if Cost^* is always the cost of an optimal plan for π. However, if we take solutions for π found by the competing planners into account when computing $\mathit{Cost}^*(\pi)$, *sat* does not satisfy IIA anymore.

We show this claim with the small example in Table 1. The leftmost table shows the cost of the reference plan R and the cost of the plans that planners A, B and C find for two tasks π_1 and π_2. If only A and B participate in the competition, A achieves a higher *sat* score than B (middle table). However, if C enters the competition as well, B is ranked higher than A (rightmost table).

Cost	R	A	B	C
π_1	2	5	4	5
π_2	6	4	5	1

sat	A	B
π_1	2/5	2/4
π_2	4/4	4/5
$\sum$	1.4	1.3

sat	A	B	C
π_1	2/5	2/4	2/5
π_2	1/4	1/5	1/1
$\sum$	0.65	0.7	1.4

Table 1: Example showing that the evaluation metric of the IPC sequential satisficing track does not satisfy independence of irrelevant alternatives if the reference plans (R) are suboptimal.

To mitigate this problem, it is important to use domain-specific solvers to find reference plans of high quality.

IPC Agile Track

IPC 2014 introduced the *agile* track (Vallati, Chrpa, and McCluskey 2014). It ignores solution costs and evaluates planners solely by how fast they are able to find a solution. The first agile competition used the following evaluation metric, which we call $\mathit{agl}_{\mathit{2014}}$: for each task π in the benchmark set that a planner *P* solves in under five minutes, *P* gets a score of $1/(1+\log_{10}(T/T^*))$, where T is the time *P* needs for solving π and T^* is the minimum runtime of any participating planner. As in the satisficing track, the total score for a planner is the sum of its scores over all tasks.

Clearly, $\mathit{agl}_{\mathit{2014}}$ does not satisfy IIA, which is the reason the agile track used a different evaluation metric in 2018. The 2018 metric, which we call $\mathit{agl}_{\mathit{2018}}$, evaluates each planner on its own. If T is the time in seconds a planner *P* needs to solve task π,[1] *P* gets the score $\mathit{agl}_{\mathit{2018}}(P,\pi)$, which is defined as

$$\mathit{agl}_{\mathit{2018}}(P,\pi) = \begin{cases} 0 & \text{if } T > 300 \\ 1 & \text{if } T < 1 \\ 1 - \frac{\log(T)}{\log(300)} & \text{if } 1 \leq T \leq 300 \end{cases}$$

[1] We define T to be ∞ if the planner exceeds the memory limit.

It is easy to see that agl_{2018} indeed satisfies IIA and is therefore preferable to agl_{2014} in our opinion.

Sparkle Planning Challenge

In 2019, the Sparkle Planning Challenge will be held for the first time. Its "primary goal [...] is to analyse the contribution of each planner to the real state of the art".[2] Consequently, the challenge measures the marginal contribution of each participating planner to a portfolio selector, i.e., an algorithm that chooses a single planner from the set of all participating planners online for a given task. In essence, the Sparkle evaluation metric, which we call *sparkle*, evaluates a planner *P* by the penalized average runtime the portfolio selector achieves when it can select from all competing planners except *P*.

The following example shows that *sparkle* does not satify IIA. Assume planners A and B enter the Sparkle Planning Challenge and the benchmark set consists of 100 tasks. Planner A solves a single task π within the time and memory limits while planner B solves the other 99 tasks but fails to solve π. Clearly, B wins the competition. Now imagine that planner C also participates in the challenge. Planner C solves the same tasks as B and has almost the same runtimes. Under *sparkle*, B and C contribute almost nothing to the portfolio (since without B the selector can still choose C and vice versa) and planner A wins the challenge.

We believe there is no reason for penalizing planners B and C for performing similarly. In fact, we consider *sparkle* to be quite problematic, since a scenario similar to the one in the example above is quite likely to come up in competitions and we see mainly two reasons for why it might.

First, the evaluation metric is easily gameable. If a team wants to win the challenge, it just needs to guess which other planners might be participating and submit similar planners in addition to the "real" planner. By the competition rules, a team of three can submit one real planner and 6 "fake" planners. Having the leader board available online before the submission deadline and the IPC 2018 planners readily available makes the metric easy to exploit.

Second, and we think this reason is even more important than the first one, the evaluation metric penalizes collaboration. It favors developing closed-source planning systems over developing planning systems openly and allowing others to build on the system. This is the case, since all planners that share the same base planning system are likely to perform well on similar tasks and therefore receive low marginal contribution scores. Openly developed planning systems have a disadvantage in the IPC as well. However, under IPC metrics a planner that builds on an open planning system might score slightly higher than the system, but under the Sparkle Challenge metric both would get a very low score.

An evaluation metric that satisfies IIA does not suffer from these problems. In order to let *sparkle* satisfy IIA, we only need to change it slightly. Instead of evaluating a planner with respect to all participating planners, we can evaluate it with respect to a fixed set of baseline planners. We believe that this set of baseline planners should be accumulated over time. A reasonable first set of baseline planners could be the set of all planners from the last IPC agile track. Subsequent challenges should add planners from later agile tracks and/or Sparkle planning challenges.

[2] http://ada.liacs.nl/events/sparkle-planning-19/

Acknowledgments

We thank the anonymous reviewers for their helpful comments. We have received funding for this work from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 817639).

References

Arrow, K. J. 1950. A difficulty in the concept of social welfare. *Journal of Political Economy* 58(4):328–346.

Vallati, M.; Chrpa, L.; and McCluskey, T. L., eds. 2014. *The Eighth International Planning Competition: Description of Participant Planners of the Deterministic Track*.

Democratizing Usage of Planning Systems by Facilitating Research in Algorithm Selection for Planning (Discussion Topic)

Michael Katz
IBM Research
Yorktown Heights, NY, USA
michael.katz1@ibm.om

Silvan Sievers
University of Basel
Basel, Switzerland
silvan.sievers@unibas.ch

Planning research has produced a large number of tools for various formalisms. As planning is a computationally challenging task, it is important to come up with a variety of ideas and approaches to tackle the various sources of planning tasks' complexity. However, as a by-product, it is unclear even to an experienced planning researcher what tool will work well on a new planning task. The challenge is even harder for a layman. Most planning tools are not easily accessible and those that are might have inadequate performance on some tasks. The problem was partially addressed by the most recent International Planning Competition, with the competing planners being made publicly available in Singularity containers, allowing for easily building and running the planners. This, however, does not solve the problem of choosing the right planner for a given task. An impatient user might forgo the option of using domain-independent planners altogether as a result of an inadequate performance of one randomly chosen planner.

Online algorithm selection using machine learning techniques was shown to be able to produce planners that show good performance on previously unseen domains. For optimal planning, previous success stories were mostly exploiting the fact that most tasks, if solved, are solved quickly. Thus, being able to accurately predict planner performance opens new perspectives for better exploiting various existing planners in practice. The research in this field does, however, currently require a rather deep familiarity with the field of planning.

In order to allow researchers from outside of the planning community to tackle the problem of planner performance prediction, we should alleviate the dependence on special knowledge in planning. One step towards achieving this goal is to provide data consumable by machine learning tools and complying with their assumptions made. The data consists of data points that each represents a planning task. Labels of each data point could represent performance of some planner on that task and features of the task, to help the user determining which features allow which planners to perform well on the task.

The most important assumption about data in machine learning is independent and identical distribution. Such an assumption is unrealistic when planning domains are created manually. Another assumption is that the data is representative of the entire population. In domain-independent planning, where the population consists of all tasks representable in the language of choice (e.g., PDDL), creating planning domains manually cannot produce representable data. For both assumptions to be satisfied, it is required to produce data automatically, and in a way that will cover a variety of possible planning tasks.

We propose a new track at IPC to help achieving the mentioned goals. The track will provide an easy access to existing planners, as well as to the data – a variety of hand-crafted, as well as automatically generated planning tasks, with additional information on the performance of these planners on the existing tasks. If providing this information seems to be unrealistic (e.g., due to computational load, as all planners would need to be run under the same conditions), an alternative could be to provide the instructions of how to obtain the performance information by running the planners (provided as containers as in the most recent IPC). Furthermore, instead of only providing a fixed set of benchmarks, one could also provide generators to automatically generate more data.

Participants in this track would submit domain-independent planners, possibly building on the provided planners, which would be evaluated on new domains like in the classical track of previous IPCs. In some sense, this new track would be similar to the learning track of previous IPCs, but with planners being provided alongside benchmarks, and with the goal of creating domain-independent rather than domain-dependent planners. The uniqueness of this track should be in alleviating the need for special knowledge in planning. The goal is to both achieve better exposure and to ease the use of planning tools outside of the planning community.

The Role of IPC in Setting Standards for Experimental Evaluation in Planning Research (Discussion Topic)

Michael Katz
IBM Research
Yorktown Heights, NY, USA
michael.katz1@ibm.om

Silvan Sievers
University of Basel
Basel, Switzerland
silvan.sievers@unibas.ch

Designing an experiment, specifically choosing a baseline for comparison as well as the benchmark set, that adequately evaluates the performance of a new suggested technique is a challenging task. Arguably, planner performance in a competition may affect the choice of a baseline. Further, the benchmark set choice may be biased towards the more recent domains, or, sometimes, the other way around, the most recent domains might be ignored.

We observed that the collection of domains introduced in a competition often aims at fostering research into a specific issue (e.g., general costs, conditional effects, large number of parameters) and thus the best performers in a competition, while excelling on these domains, might exhibit worse performance on other domains. This can be problematic and a misleading conclusion, given that researchers often seem to conclude from the latest IPC results what is the state of the art in planning.

We suggest to extend the experimental evaluation performed at future IPCs to include as many available domains as possible and computationally feasible (e.g., possibly a diverse subset of domains from previous IPCs). This extension could be an *additional* evaluation to keep the current setting that evaluates planners exclusively on new, unseen domains, which we still deem important to avoid domain-dependent overfitting of planners.

Furthermore, we suggest to create various suites according to task features and present additional performance information for existing relevant planners on each such suite. That might require to maintain such features, as well as label planning tasks according to these features, preferably in an automated manner. Such information can help, among other, in identifying the current state-of-the-art for specific fragments of planning, as well as the relevant benchmark set, making designing an experiment easier, mitigating possible biases.