Performance Robustness of AI Planners in the 2014 International Planning Competition

Andrea F. Bocchese [a], Chris Fawcett [b], Mauro Vallati [c], Alfonso E. Gerevini [a] and Holger H. Hoos [b]

[a] *Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Brescia, Italy*
[b] *Department of Computer Science, University of British Columbia, Vancouver, Canada*
[c] *School of Computing and Engineering, University of Huddersfield, Queensgate, Huddersfield, United Kingdom*

Abstract. Solver competitions have been used in many areas of AI to assess the current state of the art and guide future research and development. AI planning is no exception, and the International Planning Competition (IPC) has been frequently run for nearly two decades. Due to the organisational and computational burden involved in running these competitions, solvers are generally compared using a single homogeneous hardware and software environment for all competitors. To what extent does the specific choice of hardware and software environment have an effect on solver performance, and is that effect distributed equally across the competing solvers?

In this work, we use the competing planners and benchmark instance sets from the 2014 IPC to investigate these two questions. We recreate the 2014 IPC Optimal and Agile tracks on two distinct hardware environments and eight distinct software environments. We show that solver performance varies significantly based on the hardware and software environment, and that this variation is not equal for all planners. Furthermore, the observed variation is sufficient to change the competition rankings, including the top-ranked planners for some tracks.

Keywords: Automated Planning, Domain-Independent Planners, International Planning Competition, Algorithm Performance Robustness

1. Introduction

Competitions in AI are a useful focal point for researchers, help to drive forward research and development of solver algorithms, and provide incentives for widely sharing tools and benchmarks. Competitions also play a prominent role in evaluating and improving the state of the art of their particular research areas. Examples include AI planning, SAT, ASP, CSP, and machine learning [1–5]. Among those, the International Planning Competition (IPC) is one of the best-known, longest-running and most thoroughly designed competition series [6, 7]. Organised periodically since 1998, the IPC provides a good example of the impact of competition results on AI planning research, and on planning applications. While all the planning engines tested in the IPC are available to be used after the competition, top-ranked planners receive much of the attention and drive the research direction of the field in the years thereafter.

The great impact and success of top performing solvers implicitly rests on the assumption 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, the evaluation function used to assess solver performance, the way problem instances or planning domains are modelled, and the set of competitors [8–14]. Moreover, an analysis performed on the SAT competition showed that ranks of solvers are also affected by pseudo-random number seeds used in randomised solvers [15].

Interestingly, an investigation performed by Howe and Dahlman in 2002 showed that the relative (qualitative) performance of planners can vary when run using different hardware configurations [8]. However, as their work focused on identifying potential sources of performance variation, their analysis on the impact of hardware and software configuration was limited to assessing differences between two different machines having the same software configuration. In this work, we present the first thorough study of the impact of hardware and software environment choices, as well as resource limits, individually and jointly, on competition outcome. Specifically, our aim is to attempt to identify and isolate some aspects that have unequal impact on planner performance, with the hope that these will help with performing (and interpreting) future comparisons between planning engines.

We focus our analysis on two deterministic tracks of the 2014 International Planning Competition, the *Optimal* and *Agile* tracks. These two tracks provide a very interesting test-bed, as they rank competitors using nearly opposite metrics. In the *Optimal* track planner running time is of limited importance: planners are assessed according to their ability to generate optimal solution plans within the (large) given cutoff time. On the contrary, in the *Agile* track the quality of solutions is irrelevant, as planners are ranked according to their ability to quickly find a solution. The selected tracks also differ in terms of the benchmark instance sets used.

Our experimental analysis involves two hardware configurations and eight different software configurations. The software configurations include the choice of C++ compiler version, Python interpreter version and Java version. When running experiments on all possible combinations of hardware and software configurations, we also evaluate the impact of solver stochasticity and different running time and memory limits. Our results show that, in addition to verifying the well-known impact of memory and running time limits [8, 12], competition rankings can be affected by both hardware and software configurations. The source code for all planners, problem instances and domains, and all experimental results have been made publicly available. [1]

The remainder of this paper is organised as follows. First, we give some background on the 2014 International Planning Competition. Then, in Section 3, we describe potential sources of planner performance variation. Section 4 describes the experiment design used in our work. We present and discuss our experimental results in Section 5 and then conclude with a brief discussion of the effect of our results on the IPC.

2. The International Planning Competition

Automated planning studies the problem of finding a totally or partially ordered sequence of actions that transform a given problem environment from an initial state to a goal state (of which there may be several) [16]. Actions are usually expressed in terms of preconditions and effects. Preconditions indicate the requirements that must hold to apply an action, while effects are the consequence (including the cost) of applying the action to modify the state of the world.

The International Planning Competition has been organised since 1998, with the aims of fostering the development and comparison of planning approaches, assessing the current state of the art in planning, and identifying new and challenging benchmarks. In this paper we focus on the eighth edition of the IPC, held in 2014. For a summary of the history of previous IPCs, the interested reader is referred to López et al. [17].

IPC 2014 was held in three distinct parts: the deterministic part focused on fully observable environments where actions are atomic with deterministic effects and planning is episodic, with the presence of action costs, negative preconditions and conditional effects; the learning part, which relaxes the episodic assumption to allow planners to learn from prior experience; and the probabilistic part, with stochastic transitions and partial observability.

The deterministic part is the longest running part of IPC, and is the part that traditionally has the highest number of participants (67 in IPC 2014). Hereinafter we will focus on this part. Among the five tracks of the deterministic part of IPC 2014, here we consider the *Agile* (15 participants) and *Optimal* (17) tracks. The *Agile* track was introduced in 2014, while the *Optimal* track is among the longest-standing in the IPC.

The set of benchmarks used in the *Agile* track includes the following 14 domains: Barman, Cave Diving, Child-Snack, CityCar, Floortile, GED, Hiking, Maintenance, Openstacks, Parking, Tetris, Thoughtful, Transport, and Visitall. In the *Optimal* track, the Tidybot domain has been used in place of Thoughtful. For each track, 20 instances per domain were selected following a specifically-designed protocol [18].

In the *Optimal* track, *SymBA-2*, which is based on a symbolic bidirectional blind search with perimeter abstraction heuristics, was declared the winner and *cGamer-bd*, a bidirectional symbolic search approach that extends the *Gamer* planner (winner of the corresponding track in IPC 2008), was declared as the runner-up. Finally, *Yahsp3*, which performs a search embedding delete-relaxed heuristics, was declared as the winner of the *Agile* track of IPC 2014 and *Madagascar-pC*, which exploits a SAT-based approach to planning, was declared the runner-up.

[1] https://www.cs.ubc.ca/labs/beta/Projects/CompetitionPerformanceSensitivity

For more information about the competition, including complete results, source code of planning systems, and domain models, the interested reader is referred to the analysis of the IPC 2014 results [18], and to the official competition website.[2] Detailed descriptions of the planning systems can be found in the IPC 2014 booklet [19].

3. Sources of performance variation

When performing empirical analyses or comparisons, there are many potential influences on software performance variation. We attempt below to introduce as many such sources as possible, although we acknowledge that a full cataloguing is impossible. We investigate a subset of these sources in this work, but include all of them as a contribution and reference for future work.

3.1. Solver randomisation and other stochastic effects

Many solvers take advantage of randomisation to improve average-case performance and to avoid manual deterministic development choices. This randomisation can result in very different solver trajectories in repeated runs with different random seeds, with a correspondingly wide variation in the resulting performance. In satisfiability and other domains, empirical results demonstrate that the running time to find a valid solution is often exponentially distributed for randomised solvers [8, 20–23]. Even in the impossible case of holding everything constant in the execution environment other than random seeds, solver stochasticity would cause repeated runs to differ significantly in performance. This has also been shown empirically by Hurley and O'Sullivan [15].

Other stochastic effects on solver performance can come from the use of shared machines for experiment execution, such as large compute clusters or virtualised commodity environments such as Amazon EC2. CPU core allocation also has effects, as cache connections can vary depending on the core assignment in modern CPUs. Finally, even with no other jobs running on the same machine the operating system can (and will) context switch an experiment process in the middle of execution, causing variance in measurements of running time.

3.2. Running time and memory limits

Generally, allocating more running time or memory to solver executions will result in more problem instances solved. However, this improved performance with increased limits tends to not be distributed evenly across all solvers. For example, solvers with extensive caching or precomputation (including use of pattern databases) may benefit from increased memory limits more than other solvers. Most solver competitions, including those in automated planning, evaluate competitors with fixed limits for running time and memory. There has historically been little investigation into competitor performance outside of these limits, and the 4 GB limit used in the IPC is now less than that available in many commodity laptop computers.[3] We perform an investigation in this work studying how the benefits of higher or lower limits are distributed among competing solvers.

3.3. Hardware architecture

It is clear that hardware choices such as the CPU can affect solver performance. However, there are many choices differentiating the hardware environment of different machines, and CPU clock speed is no longer the primary source of increased performance. Performance differences can also come from CPU cache levels, processor architecture, memory bandwidth, local storage medium, network interconnection where applicable, and more. We are not aware of any solver competitions measuring competitor performance across several different machine configurations, and in this work we make a small step toward this goal by evaluating solvers on two distinct compute clusters.

3.4. Software architecture

In addition to hardware configuration, there are many aspects of the software environment configuration that can affect solver performance. These choices include the operating system used, system library options and versions (e.g. LIBC), and the compilation toolchain used for building and linking a given solver and all of its dependencies. Furthermore, there can be performance differences based on the interpreter version and configuration settings for interpreted or JIT-compiled languages like Python or Java. We investigate the effects from 8 different software configurations in this work.

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

[3] This limit has been increased to 8 GB for IPC 2018.

3.5. Choice of benchmark distribution

Evaluating solver performance requires runs on one or more problem instances forming a benchmark set. (In planning, these instances are additionally from distinct planning domains.) Benchmarks should be challenging for the participating solvers, and need to allow for performance differences between solvers to be identified. In a nutshell, benchmarks must be neither too challenging, where no solver is able to provide any solution within the given resource limits, nor trivially solvable: in both cases, no differences between solvers can be identified. In some areas of AI the complexity of problems can be evaluated statistically, without running any solver, by considering the phase transition [24, 25]. This is usually not the case in planning. A phase transition has been demonstrated for randomly-generated graphs [26, 27], but is typically unknown on instances from newly-designed domain models. However, recent work by Cohen and Beck [21] provide an empirical investigation of the phase transition phenomena for heuristic search, focusing on the exploitation of greedy best first search. In fact, the difficulty of planning problems has been mainly assessed experimentally, i.e., by running solvers. For instance, the recently introduced Torchlight tool [28] allows planner developers and users to analyse the search space topology of planning problems under the delete-relaxation heuristic.

Beside the issue of assessing problem instance complexity, planning instances are often created using randomised generators, where a few parameters define the size and the complexity of the resulting instances. The choice of problem instance domains, randomised generator settings, and instance set size and distribution will all have an (uneven) effect on competing solver performance. We consider this source of variation out of scope for this work, and focus solely on the official benchmark instances used in the 2014 IPC. A discussion on protocols for benchmark selection in planning competitions is provided by Vallati and Vaquero [14], and recommendations on benchmark selections were also provided by Howe and Dahlman [8].

3.6. Choice of performance aggregation and ranking mechanism

Given a set of solvers and benchmark problem instances, competition organisers and others interested in empirical performance evaluation must make further evaluation decisions. These decisions include how solver performance is aggregated across the set of benchmark problems, and the metric used (running time, instance set coverage, solution quality). Some competitions use an absolute scoring mechanism (such as mean running time), while others like the IPC use scoring mechanisms where each competitor can have an effect on the score of other competitors. In fact, each track of the IPC typically uses its own scoring mechanism. Tiebreaking mechanisms can also affect the final solver rankings. All of the above choices have been held constant in the IPC for some time, and while the question of whether there are qualitatively better choices is an interesting one, we consider a full treatment of this topic to be out of scope for this work. However, performing an initial analysis of these effects is useful and straightforward with existing competition data, and can help contextualise and characterise the relative impact of different scoring mechanisms on solver rankings. In section 5.4 we provide the results of such an empirical analysis, looking at the effect of alternative scoring mechanisms on the two IPC 2014 tracks considered in this paper.

4. Experiment design

For our experimental analysis, we chose two sequential, deterministic tracks of the 2014 International Planning Competition (IPC): the *Agile* and the *Optimal* tracks. These two tracks provide a very interesting test-bed, as they rank competitors using nearly opposite metrics and also differ in terms of the benchmark sets. The *Optimal* track is among 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.

In the *Agile* track, competing planners are evaluated based on the running time required to find any satisficing plan, with no regard to the quality of that plan. There were 15 competing planners in the IPC 2014 *Agile* track, evaluated on 20 benchmark instances from each of 14 planning domains (280 instances in total). These planners were given a running time limit of 300 CPU seconds, on a single CPU core, and performance was evaluated using the *IPC running time score*. For each problem instance i, let t_i^* be the minimum running time required for any competing planner to produce a satisficing plan. Any planner that successfully produces a satisficing plan in time t will receive a score

Table 1

Hardware specification of the Orcinus and Galileo computer clusters used in our experiments, and of the cluster used to run the official IPC-2014 competition. Unfortunately, the IPC-2014 cluster has been upgraded since the competition and the CPU model used at the time of the competition is unknown.

Cluster	Nodes	Total Cores	Cores/node	CPU type	CPU Ghz
Orcinus	544	6528	12	Xeon X5650	2.67
Galileo	516	8256	16	Xeon E5-2630 v3	2.40
IPC-2014	34	136	4	–	2.39
	Cores/CPU	Cache [MB]	RAM [GB/node]	OS	
Orcinus	6	12	24	CentOS 5.0	
Galileo	8	20	128	CentOS 7.0	
IPC-2014	4	12	8	Red Hat 4.4.7	

of $1/[1 + log_{10}(t/t_i^*)]$ for i. Failure to produce a satisficing plan within the CPU time limit results in a score of 0 for i. If t is less than 1 CPU second, the score is set to 1, to prevent large score differences on trivial instances. The final score for each competing planner is the sum of the scores for that planner over all instances i of the benchmark set.

In the *Optimal* track, competing planners are evaluated based on the ability to find an optimal-cost plan. There were 17 planners competing in the *Optimal* track of IPC 2014, again evaluated on 280 benchmark instances (20 instances from each of 14 domains). These planners were given 30 CPU minutes of running time, on a single CPU core. The running time required to produce this plan plays no role in scoring, and planners are simply assigned a score of 1 if an optimal plan was produced for instance i, and 0 otherwise. As for the *Agile* track, the final score for each competing planner is defined as the sum of the individual instance scores.

Many of the planners from the selected tracks required 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. The planners requiring the most modifications were *cGamer-bd*, *DPMPlan*, *MIPlan* and *NuCeLaR*; these planners all use the same parser for grounded PDDL, which writes files into the directory containing the planner's source code. In order to adapt the parser to our environment and support runs performed in parallel, we modified the source code to instead write these files into the planner's working directory. An analogous solution was applied to *RIDA*, where the planner also dynamically wrote files into its own source directory [19].

We consider these modifications minor and do not believe that they had any effect on planner execution or running times. There were two planners that could not be made to work in our environments, 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 either of the computer clusters or with any version of Java at our disposal. 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. Both planners were ranked outside of the top 5 planners in their respective tracks, placing approximately in the middle of the competition rankings. We fully expect that if these issues were to be fixed, they would not significantly change our data or conclusions.

In order to investigate the performance effects of hardware architecture, we utilized two large compute clusters: the Compute-Calcul Canada WestGrid *Orcinus* cluster [4] and the Italian CINECA *Galileo* cluster. [5] More details on the hardware and software configuration of these clusters are given in Table 1. Hyperthreading was disabled on both clusters. We note that both

[4] https://www.westgrid.ca/support/systems/orcinus

[5] http://www.hpc.cineca.it/hardware/galileo

Orcinus and *Galileo* have more (and more powerful) CPU cores than those of the cluster used for running the IPC 2014. It is also noticeable that they share the same OS, though in different versions. Both the hardware architecture and the OS are elements that are often beyond our control, but can still have an impact on the performance of solvers, as we demonstrate in this work.

Due to the significant resource requirements of reproducing the IPC competition results, we were limited to these two clusters, as we had existing large resource allocation grants for both. We expect that the variance results in this paper will be similar or more significant on other hardware architectures, especially those based on CPUs other than the newer Intel Xeon chips utilised in Orcinus and Galileo.

For the analysis of performance variation over different software configurations, we chose to investigate three major software components: GCC compiler version, Python interpreter version, and Java version. Nearly every planner 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 do not successfully compile with versions of GCC 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 on which all relevant planners would execute successfully. The combination of these options resulted in 8 potential software configurations, all of which were used in this work. We will frequently refer in our results to the configuration provided as default in IPC 2014 (GCC 4.7.2, Python 2.7.3, Java 1.7.0) as the *base* configuration, and the configuration with the most recent of each option (GCC 4.8.2, Python 2.7.10, Java 1.8.0) as the *newest* configuration. However, it should be noted that IPC 2014 participants were allowed to require a specific version of software dependencies to be used for running their planner.

We then evaluated all considered planners from each track on the entire competition benchmark sets, for each of the 16 (*hardware*, *GCC version*, *Python version*, *Java version*) configuration options. In order to account for and measure solver stochasticity, we performed 5 independent runs of each configuration. This resulted in 80 complete reproductions of the IPC 2014 *Agile* and *Optimal* tracks. All planner runs were performed independently in parallel, with each run assigned 1 CPU core, 8 GB of RAM, and a running time limit of 1800 (Optimal track) or 300 (Agile track) CPU seconds. These running time limits are the same as those used in the IPC tracks, but the competition memory limit was only 4 GB.

We have used a memory limit of 8 GB in this paper, primarily to offset the increased memory usage when forcing compilation for 64-bit execution. As our focus in this work is the impact of hardware and software configuration on planner performance, we also wanted to avoid memory limits being exceeded as much as possible. We use our results with an 8 GB memory limit to produce hallucinated results using a 4 GB limit, as follows: for each considered planner run, if the 8 GB data shows a peak memory usage for that planner higher than 4 GB, that problem instance is counted as unsolved for that planner in the hallucinated data. A set of experiments performed using hard 4 GB limits (described in Section 5.2) indicates that these hallucinated results are consistent with those obtained by setting hard RAM limits.

All planners were explicitly compiled for 64-bit execution, as neither of our clusters have support for 32-bit execution. The cluster on which IPC 2014 was run had a 64-bit architecture, however competitors were allowed to require their planners to be compiled and executed as 32-bit. Running time and memory limits in our experiments were monitored and enforced using tools from AClib. [29]. [6] These tools are built on top of standard system tools such as ulimit, and the limit enforcement is consistent with that used in the IPC.

5. Results

This section is devoted to the empirical evaluation of the influences of sources of performance variation on the results of the agile and optimal tracks of IPC 2014. In many of our results, we make use of so-called *bump charts* to graphically represent the performance variation of our considered planners across several hardware and software configuration options; an example is shown in Figure 1. In these charts, each vertical “column” represents the performance of a different set of runs of our considered planners. The points for each planner are connected and coloured to better illustrate the performance differences between configurations.

[6] http://www.aclib.net

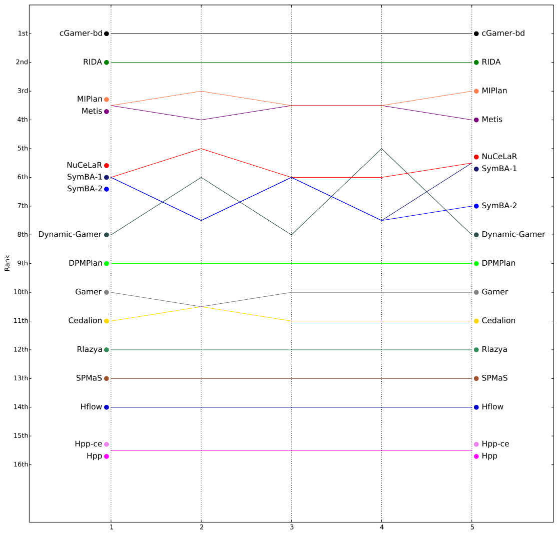


Fig. 1. Competition ranking among different runs of the *Optimal* track with the same *"base"* configuration on *Orcinus*.

Table 2

Coverage of different runs on the same hardware (*Orcinus*) with the same *"base"* configuration, for the *Optimal* track.

	Instance coverage						
Planner	#1	#2	#3	#4	#5	average	σ^2
cGamer-bd	128	131	131	130	131	130.2	1.7
RIDA	116	118	117	118	117	117.2	0.7
MIPlan	112	113	112	112	114	112.6	0.8
Metis	112	112	112	112	113	112.2	0.2
NuCeLaR	108	110	108	109	109	108.8	0.7
SymBA-1	108	108	108	108	109	108.2	0.2
SymBA-2	108	108	108	108	108	108.0	0.0
Dynamic-Gamer	105	109	107	110	107	107.6	3.8
DPMPlan	100	101	102	102	102	101.4	0.8
Gamer	96	95	98	98	97	96.8	1.7
Cedalion	95	95	95	96	96	95.4	0.3
Rlazya	91	91	92	92	89	91.0	1.5
SPMaS	81	74	74	73	77	75.8	10.7
Hflow	56	56	56	56	56	56.0	0.0
Hpp	15	15	15	15	15	15.0	0.0
Hpp-ce	15	15	15	15	15	15.0	0.0

5.1. *The effects of solver randomisation*

It is common practice to include randomised components in AI planning systems. Randomisation is useful, e.g., for breaking ties during the heuristic search process, introducing some noise in the heuristic search state evaluation, and for diversification when performing search restarts. Evidently, solver performance can be affected by this source of stochasticity.

In order to quantify this variability, we examine five independent runs of all the participants of the competition tracks considered in our study, on the same platform, in this case the *"base"* configuration of *Orcinus*. For planners that allowed it, we fixed the seed parameter used for the randomised component. The underlying assumption is that this performance variability is orthogonal to hardware and software configuration choices, and the results for the other 15 configurations are indeed very similar.

Figure 1 shows the variation in competition rank of planners that took part in the optimal track, for each of the 5 independent runs. Table 2 presents the corresponding numerical values in terms of instance coverage, along with the performance variance for each planner.[7] Very few planners of the *Optimal* track show significant variability in terms of coverage. *SPMaS* and *Dynamic-Gamer* are the planners that show the largest difference in terms of instances solved within the allotted time; most of the planners show a discrepancy of only 1 or 2 instances.

A similar picture emerges when the performances of the *Agile* track planners are analysed. In fact, these planners show less variation in terms of instance coverage. However, in the IPC *Agile* track, planners have been evaluated according to their IPC runtime score. From this perspective, *ArvandHerd* and *Jasper* show the largest score fluctuations: the score of the former ranges between 94.0 and 84.8, while the IPC score of *Jasper* stays in the 90.0–81.6 range. The impact of this IPC score variation on the competition ranks is limited: in each run, at most two pairs of planners swapped their ranks. However, these results do show that the ranks can change in repeated runs.

For both the considered clusters, we were allowed to reserve cores –with the corresponding amount of dedicated RAM– for our experiments, hence minimising the variability due to having different processes sharing such resources. We are confident that to the extent possible without deep planner source code modification, we isolated the impact of planner randomisation and reduced the impact of variance due to hardware and software factors not considered in this work.

These experimental results confirm that, while the performance variation of these planners due to stochasticity tends to be limited, it cannot be ignored. For this reason, in order to try to correctly account for stochasticity when assessing the impact of the different sources of variation in the following subsections, the

[7] Table caption: Coverage of different runs on the same hardware (*Orcinus*) with the same *"base"* configuration, for the *Optimal* track. We used this footnote as the provided tex style seems to have issues with captions of tables.

presented results will be derived by considering average performance over the five independent runs per instance. However, it is important to mention that there is still a possibility that a portion of the planner performance variation observed in our experiments is due to stochastic noise that has not been removed by considering the average of multiple runs.

5.2. *The effects of memory limits*

It is well known that the amount of RAM available for a planner has a strong impact on its performance [8, 12]. In addition to providing further confirmation of this previous work, in this section we are interested in investigating if this source of variation similarly affects all of the planners considered in our study. To investigate this, we considered the five independent runs of our *"base"* configuration on *Orcinus*. The memory limit used for these runs was 8 GB, as with the other experiments in this work. We recorded the peak memory usage for each run, which we used to hallucinate the result of running each planner with a memory limit lower than 8 GB. This approach does not work for planners that pre-allocate resources to fill their memory allocation, but in practice, we did not see this behaviour in the planners we studied. We performed an additional set of runs with an explicit 4 GB RAM limit to test the effect of hallucinating lower memory limits, and planner performance was very similar to the hallucinated predictions (with the exception of some planners from the *Gamer* family, discussed further below).

Figure 2 shows the hallucinated cumulative coverage of the *Optimal* track planners with respect to the available amount of RAM. Interestingly, most of the planners show a significant performance improvement when the amount of available RAM ranges between 4 and 5 GB. Moreover, planners based on Java –from the *Gamer* family– show a very peculiar behaviour: almost no solutions are found when less than 3 GB of RAM are available. The effect of the Java Virtual Machine on memory consumption can be clearly observed by looking at the performance of *NuCeLaR*, a portfolio planner that exploits *Gamer* as a basic solver. In fact, we observed that, when manually configured by using the *ms* and *mx* JVM parameters, the RAM requirements of the planners that use Java can be significantly reduced.

Hallucinated instance set coverage of planners that took part in the *Agile* track of IPC 2014 were similar to those discussed above: Java-based planners required at least 3 GB of RAM in order to solve any instance. However, one difference between the *Agile* and *Optimal* planners was that the coverage of the *Agile* planners was not typically improved when more than 4 GB of RAM were available. Exceptions were only *Madagascar-p* and *Yahsp3-mt*, which were able to exploit the higher memory limit to solve more instances.

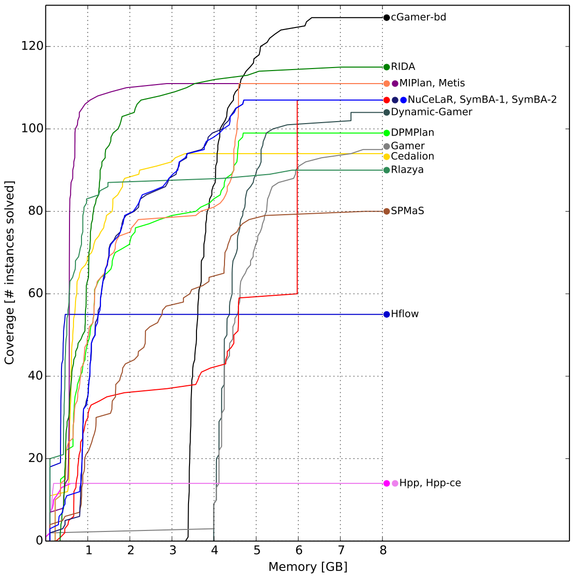


Fig. 2. Cumulative distribution function of the coverage over the RAM consumption for the solved instance in the *Optimal* track on *Orcinus* (*"base"* configuration).

Finally, by examining the performance of Java-based planners across our Java 1.7 and Java 1.8 configurations, we conclude that the latter typically forces planners to use a larger amount of RAM, on average around 1 GB more.

5.3. *The effects of running time limits*

It comes as no surprise that increasing or decreasing the available running time has an impact on the performance of many planners. However, it is also a common belief in the literature that most classical planners *either solve a problem quickly or not at all within reasonable running time* [8, 30]. In this section, we aim to investigate this hypothesis, as well as to examine whether the performance differences resulting from different running time limits are evenly distributed across planners.

Figure 3 shows the cumulative number of solved instances for planners that took part in the IPC 2014 *Optimal* track. Experiments were run on our *Orcinus* hardware configuration, using our *"base"* software configuration. These results indicate that many of the ranks do not change when the cutoff time is higher than

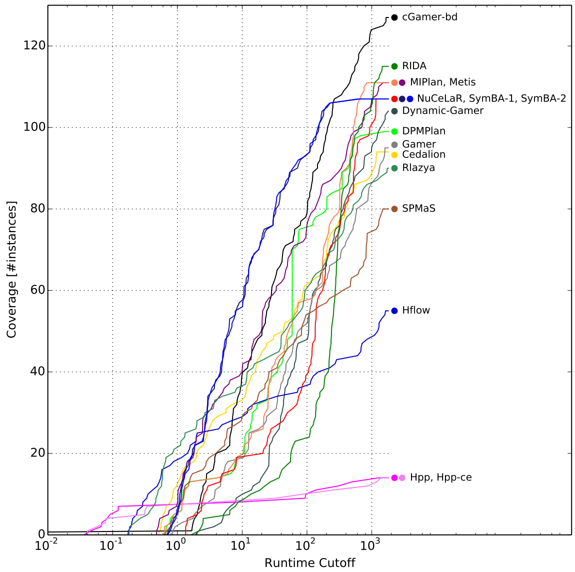


Fig. 3. Cumulative distribution function of the coverage over runtime for the *Optimal* track on *Orcinus* (*"base"* configuration).

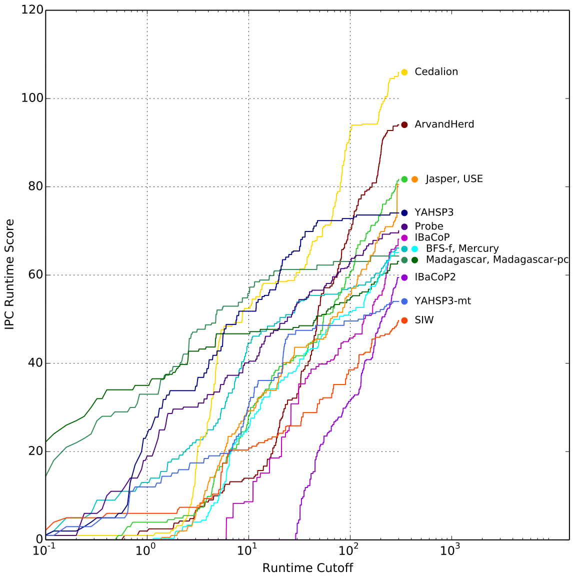


Fig. 4. Cumulative distribution function of the IPC runtime score over runtime for the *Agile* track on *Orcinus* (*"base"* configuration).

10 seconds. Moreover, after 10 seconds many of the planners continue to solve additional instances as the available running time increases. However, there are a number of exceptions. *DPMPlan* solves a significant number of instances using approximately 70–80 seconds: analyses indicate that this is due to the fact that this planner uses a sequential portfolio, and around that running time a new planner is typically started. *RIDA* solves the vast majority of its instances using more than 120 seconds while, on the contrary, the *SymBA-1* and *SymBA-2* planners do not solve any additional instances using running time cutoffs of more than 100 seconds.

The *Agile* track planners show a similar overall behaviour in terms of coverage. Our analyses indicate that increasing the available running time leads to an expected improvement in instance set coverage for most of the planners: only the *Yahsp3* planners show a flat cumulative coverage function before the 5 minute limit. Additionally, it seems that the competition rankings are not stable and are significantly affected by the chosen cutoff time. Figure 4 shows how the IPC runtime score of planners that took part in the IPC 2014 *Agile* track is affected by the cutoff time. As it is apparent, many ranks change also when the cutoff time is higher than 100 seconds, indicating that planners are still solving instances and improving their IPC runtime score. *Cedalion* and *ArvandHerd* provide a good example of the described behavior; their IPC runtime scores keep growing for cutoff times higher than 1 CPU-time second. On the contrary, planners like *YAHSP3* and *Madagascar* are able to solve a significant number of instances in a very short time –less than 60 seconds–, but after that their IPC runtime scores do not improve significantly, as the additional CPU-time does not allow them to solve as many additional instances.

According to the results shown in Figures 3 and 4, the past observation that planners either solve a problem quickly or fail to solve it within reasonable time does not appear to hold any longer. Most of the considered planners can utilise all of the running time provided in the competition. Remarkably, this observation not only applies to portfolio-based planners, but also to those based on a single planning approach. In terms of the relative impact on planner performance, the impact of running time differs substantially between planners. In the *Optimal* track, performance ranks do not vary significantly when more than 10 seconds are available. *Agile* track rankings, on the other hand, are strongly affected by the cutoff time.

5.4. *The effects of scoring mechanisms*

When comparing the performance of two or more planners, there are several potential alternatives to produce a ranked ordering. These options include the number of problem instances solved by each planner (instance set coverage), IPC runtime/quality scores,

Table 3

Scores computed for *Agile* track planners using several alternative scoring mechanisms. Planners are ordered by their IPC runtime score, the choice used in the IPC 2014 competition. Higher instance set coverage and IPC runtime scores are better, and lower PAR-1 and PAR-10 scores are better.

		Scoring Method		
Planner	Coverage	IPC runtime	PAR-1	PAR-10
Cedalion	159.4	107.50	161.86	1324.78
ArvandHerd	149.2	89.24	189.96	1451.24
Jasper	140.0	84.84	183.31	1533.31
USE	149.0	79.36	189.44	1452.66
YAHSP3	87.8	73.75	211.17	2064.53
IBaCoP	159.0	71.53	183.68	1350.47
Probe	106.2	71.44	201.64	1877.57
Mercury	125.4	68.21	196.19	1686.98
BFS-f	109.6	66.18	204.40	1847.15
Madagascar	75.0	64.11	221.92	2198.71
Madagascar-pc	87.4	63.43	221.62	2078.84
IBaCoP2	153.8	60.56	196.24	1413.17
YAHSP3-mt	76.8	56.27	228.28	2187.70
SIW	67.2	49.73	246.64	2298.64

Table 4

Scores computed for *Optimal* track planners using several alternative scoring mechanisms. Planners are ordered by their instance set coverage, the choice used in the IPC 2014 competition. Higher instance set coverage and IPC runtime scores are better, and lower PAR-1 and PAR-10 scores are better.

		Scoring Method		
Planner	Coverage	IPC runtime	PAR-1	PAR-10
cGamer-bd	130.2	96.93	1048.91	9715.91
RIDA	117.2	53.91	1191.54	10610.69
MIPlan	112.6	58.99	1152.89	10838.17
Metis	112.2	77.12	1148.01	10856.44
NuCeLaR	108.8	51.88	1199.83	11104.97
SymBA-1	108.2	89.02	1118.69	11058.55
SymBA-2	108.0	89.29	1118.67	11070.09
Dynamic-Gamer	107.6	59.94	1217.66	11192.24
DPMPlan	101.4	59.15	1188.68	11521.97
Gamer	96.8	49.84	1280.37	11879.80
Cedalion	95.4	57.25	1246.63	11927.05
Rlazya	91.0	63.41	1272.49	12207.49
SPMaS	75.8	47.44	1372.67	13187.10
Hflow	56.0	40.98	1490.98	14450.98
Hpp-ce	15.0	10.30	1716.27	17048.41
Hpp	15.0	10.14	1711.79	17043.94

and penalised average runtime (PAR) scores. PAR-10 (PAR-1) is a metric often used in automated algorithm design and empirical analysis experiments, where average runtime is modified by counting runs that did not find a plan as ten (one) times the running time cutoff. The choice of a specific scoring mechanism incentivises competitors to optimise their submissions with respect to that mechanism, and therefore this choice can affect the resulting rankings.

In order to investigate the effect of scoring mechanisms on planner rankings, we computed instance set coverage, IPC runtime score, PAR-1 and PAR-10 scores using the independent *Agile* and *Optimal* track runs gathered on the *Orcinus base* environment configuration.

Table 3 summarises the various scoring mechanisms for the *Agile* track. The scoring mechanism used in the competition was the IPC runtime score. This metric penalises planners for failing to solve problem instances solved by other planners, and ignores timing differences between planners successfully solving a problem instance in less than 1 CPU second. Planners such as IBaCoP2 appear to have solved almost the same number of problem instances as the winning planners, but with running times that reduced their IPC runtime score substantially. The PAR-1 and PAR-10 scores appear to be a compromise between instance set coverage and IPC runtime score.

Table 4 summarises the scoring results for the *Optimal* track. In this case, the competition scoring metric was strictly instance set coverage over a 1800 CPU second running time cutoff. When the running time to produce an optimal solution is taken into account *cGamer-bd* is still the top-performing planner, but there are changes in many of the other ranks. For example, the performance of *Metis* and the *SymBA* planners improves significantly, and the *RIDA*, *MIPlan* and *NuCeLaR* planners see a decrease.

5.5. *The effects of hardware architecture configuration*

Previous work in this area (such as that of Howe and Dahlman [8]) has presented evidence that the hardware platform used can influence planner performance. Different CPU-clock speeds can behave like a different running time cutoff, different amounts of RAM can change the problems that can be solved by one algorithm significantly, and the architecture design of the CPU and other factors can influence performance in a manner that is harder to predict. What we want to analyse here is the *relative* performance changes between planners from these factors, since this can lead to a different competition ranking based on the specific hardware configuration chosen. Here we use the *Orcinus* and *Galileo* clusters as our two considered hardware configurations, and we also investigate the differences between the performance on our clusters and the official IPC 2014 results. While this analysis does make every effort to isolate the effects of the hardware configuration alone, some software influences are still present since the two clusters run different operating system versions and several system libraries out of our control are not identical.

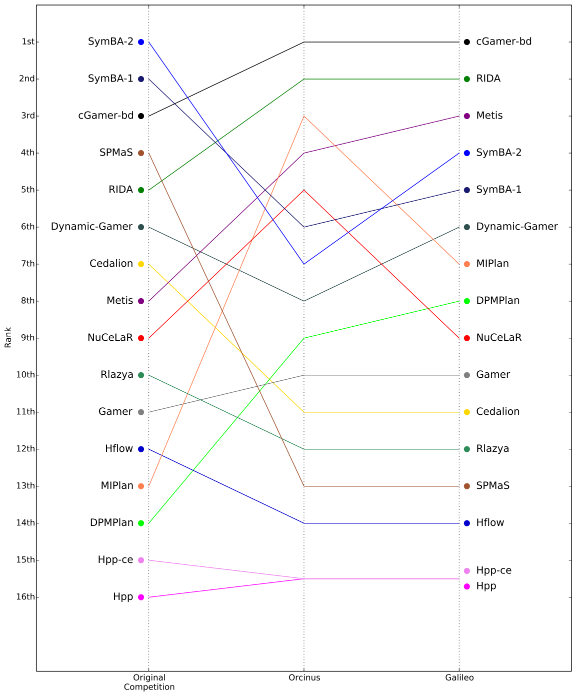


Fig. 5. Comparison of the ranking using our two hardware configurations (*"base"* software configuration), for the IPC 2014 *Optimal* track. We also show the official IPC 2014 results.

5.5.1. *Differences between hardware configurations*

Looking at the second and third column of Figures 5 and 6, which respectively represent the results for *Orcinus* and *Galileo* on each of the considered IPC 2014 tracks, we can see a mixed set of trends. The instance coverage and IPC runtime score for each track, respectively, are given in Tables 5 and 6. In the *Optimal* track, the trend is largely neutral to positive, with most planners obtaining equal or slightly-higher instance set coverage on *Galileo* than *Orcinus*. However, there are three exceptions: *MIPlan*, *NuCeLaR*, and to a lesser extent *SPMaS*. These trends are more significant for the *Agile* track due to the use of the IPC runtime score, with rank changes occurring from both performance improvement (*Jasper*, *Mercury*, *BFS-f*, *Madagascar-pc*, *SIW*) and performance degradation (*ArvandHerd*, *Probe*, *Madagascar*, *Yahsp3-mt*) between *Orcinus* and *Galileo*.

We believe that the cause of the mostly-positive trends can be partially explained by the better single-core hardware performance of the *Galileo* cluster. Even though *Orcinus* has a "faster" CPU in terms of clock speed, *Galileo* has a newer hardware architecture, more available cache, and better memory bandwidth. What we consider most interesting in these experimental results are those planners that significantly deviate from the neutral-to-positive trend, and that we see both significant performance improvements for some planners and significant performance degradation for others. A full explanation for these deviations is difficult, but our observations suggest that the hardware platform does not affect all planners in the same way.

Table 5

Comparison of planner performance on our two hardware configurations (*"base"* software configuration), for the IPC 2014 *Optimal* track. We also show the official IPC 2014 results.

	Instance coverage		
Planner	IPC-2014	*Orcinus*	*Galileo*
SymBA-2	151	108	109
SymBA-1	143	108.2	108.6
cGamer-bd	120	130.2	132.3
SPMaS	114	75.8	71.3
RIDA	113	117.2	117.6
Dynamic-Gamer	99	107.6	107.6
Cedalion	93	95.4	96
Metis	91	112.2	113
NuCeLaR	90	108.8	98.3
Rlazya	88	91	92.6
Gamer	83	96.8	97
Hflow	53	56	58.3
MIPlan	47	112.6	103
DPMPlan	43	101.4	102
Hpp-ce	15	15	15
Hpp	14	15	15

5.5.2. *Comparison with IPC 2014 results*

We now turn our attention to the differences between the official results of IPC 2014 and the performance on our two hardware configurations, referring again to Figures 5, 6, along with Tables 5, and 6.

It should be noted that during the 2014 IPC competition, participating teams were allowed to require (or select) the most appropriate version of every software component needed by the submitted planner, whether to use 64- or 32-bit execution, and RAM was limited to 4 GB (while we consider an 8 GB limit). Unfortunately, the cluster used for running the competition is no longer available, as it was replaced shortly after the competition. For these reasons, a direct comparison between the results obtained on our machines and the official IPC 2014 results is not possible. However, we try to analyse here some general trends that can be observed even in a very rough comparison.

In the *Optimal* track, the performance differences causing rank changes between planners are mainly caused by a few planners with significantly different performance between IPC 2014 and our hardware configurations. The *SymBA-1*, *SymBA-2* and *SPMaS* planners show a significant performance degradation,

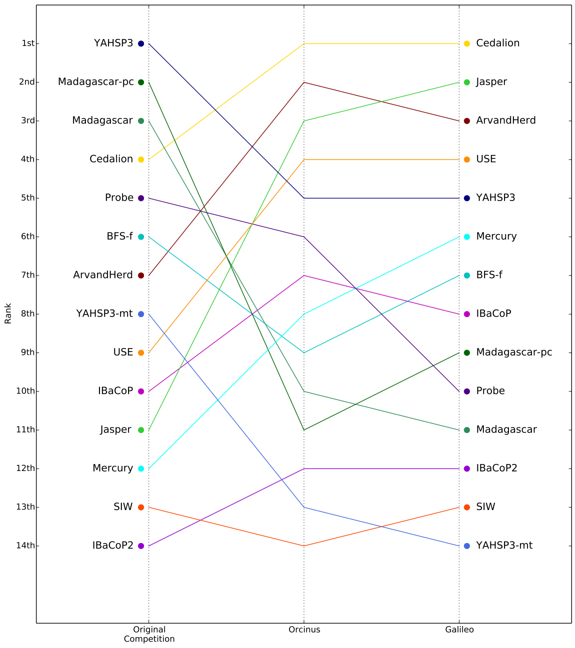


Fig. 6. Comparison of the ranking using our two hardware configurations (*"base"* software configuration), for the IPC 2014 *Agile* track. We also show the official IPC 2014 results.

Table 6

Comparison of planner performance on our two hardware configurations (*"base"* configuration), for the IPC 2014 *Agile* track. We also show the official IPC 2014 results.

	Runtime score		
Planner	IPC-2014	*Orcinus*	*Galileo*
YAHSP3	81.2	73.7	73.3
Madagascar-pc	70.5	63.4	65.6
Madagascar	65.4	64.1	63.7
Cedalion	65.3	107.4	113
Probe	65.2	71.4	65.2
BFS-f	60.9	66.1	69.6
ArvandHerd	57.2	89.2	88.4
YAHSP3-mt	51.7	56	45.4
USE	47.3	79.3	81.4
IBaCoP	47	70.1	69
Jasper	42.8	84.8	89.6
Mercury	37.5	67.8	73.1
SIW	35.1	49.6	52.6
IBaCoP2	31.7	59.6	57.3

whereas *MIPlan*, *DPMPlan*, *Metis* and *NuCeLaR* show significant performance improvements. The specific causes of these changes in each planner are unclear, but we did identify in the IPC 2014 published planner logs that *MIPlan* had several crashed runs in the competition due to a library dependency error, and that *DPMPlan* was disproportionally affected by the competition's 4 GB RAM limit.

In the *Agile* track we observe a general trend of improved IPC runtime score performance between the IPC 2014 results and our hardware configurations. As in the *Optimal* track, several planners are affected differently and exhibit significant performance degradation, in this case with the two *Madagascar* and two *Yahsp3* planners. Table 6 demonstrates that the positive performance trend is also not distributed evenly among the competing planners, which is the main cause of the ranking changes other than the planners with performance degradation. The *Cedalion* planner is the system that gains the most when run on both of our hardware configurations.

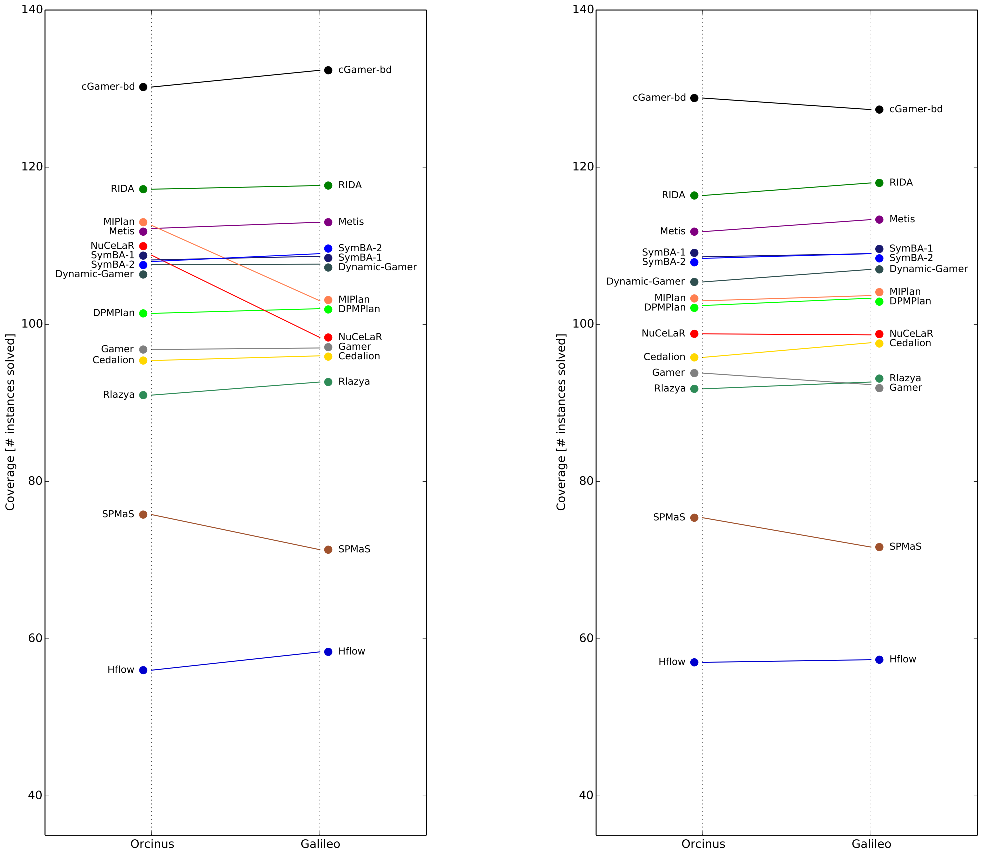
5.5.3. Hardware and software synergies

In order to better understand the role of the hardware configuration in performance variation, we repeated the previous analysis for a different software configuration. In this case, we used the *"newest"* configuration rather than our *"base"* configuration. We present side-by-side bump charts for the two analyses in Figures 7 and 8, respectively, for the *Optimal* and *Agile* tracks. Many of the planners show very similar performance changes between our hardware configurations in both scenarios, but several planners change their behaviour dramatically. In the *Optimal* track, the significant performance degradation seen for the *MIPlan* and *NuCeLaR* planners in the *"base"* configuration disappear completely for the *"newest"* configuration. In the *Agile* track, there are performance differences due to the running time changes between the two software configurations, but the trends are largely similar other than for the two *IBaCoP* planners which show opposite trends.

5.6. The effects of software architecture configuration

In order to evaluate the impact of the software environment configuration on planner performance, we examine the results of our eight software configurations using the consistent hardware configuration provided by the *Orcinus* cluster. These eight software configurations reflect the choice of GCC compiler version (4.7.2 or 4.8.2), Python interpreter version (2.7.3 or 2.7.10) and the version of the Java JDK and virtual machine (1.7 or 1.8).

Figure 9 shows how the instance set coverage (and therefore the competition ranking) of the *Optimal* track planners are affected by our software configurations. The corresponding data can be found in Table 7. It appears that a number of planners have a tangible performance drop when Java 1.8 is used instead of ver-



(a) *"base"* configuration.

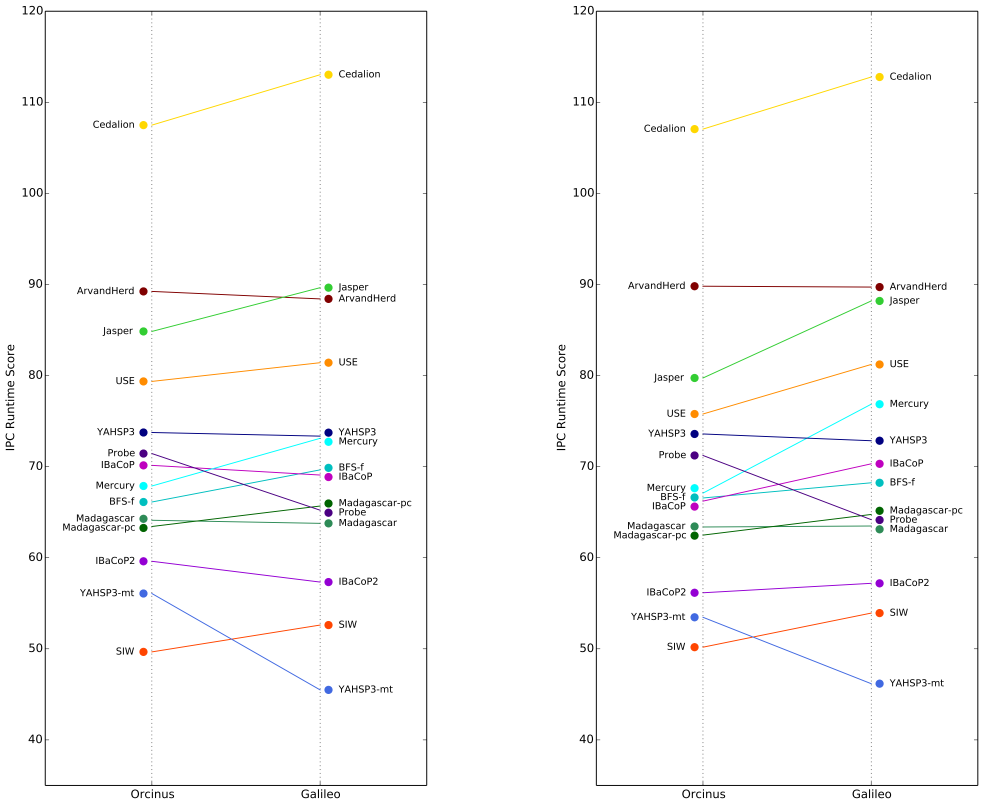
(b) *"newest"* configuration.

Fig. 7. Comparison between the *"newest"* and the *"base"* configuration hardware sensitiveness of the *Optimal* track. The *Hpp* and *Hpp-ce* planners had a constant result of 15 problem instances solved and are not shown here.

sion 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 solvers. The remaining planners of the *Optimal* track show limited, but in the case of *RIDA* and *SPMaS* still noticeable, performance fluctuation.

Figure 10 shows how the software configuration affects the instance coverage of planners from the *Agile* track. The corresponding coverage data is presented in Table 8, along with the IPC runtime scores for the sake of completeness. We focused on coverage because this allows for an objective assessment of the performance of each planner. The IPC score of each planner depends on the performance of all other planners, and may obscure performance differences in individual planners. Instead, the instance set coverage of a planner is an absolute measure that does not depend on any other planner. However, IPC score and coverage are also closely related: if a planner is unable to solve a given benchmark instance, the corresponding IPC runtime score for the instance will be 0.0.

As shown in Figure 10, some planners from the *Agile* track demonstrated a sensitivity to the GCC compiler version. For example, extreme variation can be observed in the performance of *IBaCoP* and *IBaCoP2*. The performance of the other *Agile* track planners, in particular *Use*, *ArvandHerd* and *Jasper*, are affected by a combination of GCC compiler and Python versions. However, we note that *ArvandHerd*, *Jasper* and *Yahsp3-mt*, which according to the results in Figure 10 show remarkable performance variation, are also among the planners with the highest variance on multiple runs (see Section 5.1). Therefore, there is a possibility that a portion of the observed software configuration variation is due to stochastic noise that was not removed by considering the average of five runs.



(a) *"base"* configuration.

(b) *"newest"* configuration.

Fig. 8. Comparison between the *"base"* (left) and the *"newest"* (right) configuration hardware sensitiveness of the *Agile* track.

To summarise our software configuration analyses, the specific choice of software configuration has an impact on the performance of planners, in many cases a significant impact. The impact may be the result of a number of factors, such as:

- Planner dependence on a variety of software technologies. Intuitively, this is due to the fact that a planner can then be affected by multiple changes simultaneously. This is especially true for portfolio planners, such as *NuCeLar*.
- Planners that have been highly optimised for specific versions of a particular software package are sensitive to subsequent version changes. *SymBA*, for instance, explicitly required a specific version of the GCC compiler to be installed on the IPC 2014 benchmarking environment.
- Major changes in the way in which software components work, such as Java JVM memory allocation and garbage collection from version 1.7 to 1.8, strongly affect planners relying on those components.

As impacts are not distributed evenly among all planners, the choice of specific software configuration can dramatically affect competition results.

6. Conclusions and future work

In this work we presented an empirical investigation of solver performance variation across several options for hardware and software architecture configuration. For each of our 8 software and 2 hardware configurations, we ran the planners used in the deterministic optimal and agile tracks of the 2014 International Planning Competition (IPC 2014), effectively repeating the 2014 competition multiple times, independently for each considered configuration.

Our analysis shows that the hardware and software environment has a significant effect on solver performance, and that this effect can also vary significantly for different solvers. As a result, rankings in competitions such as the IPC cannot be expected to generalise

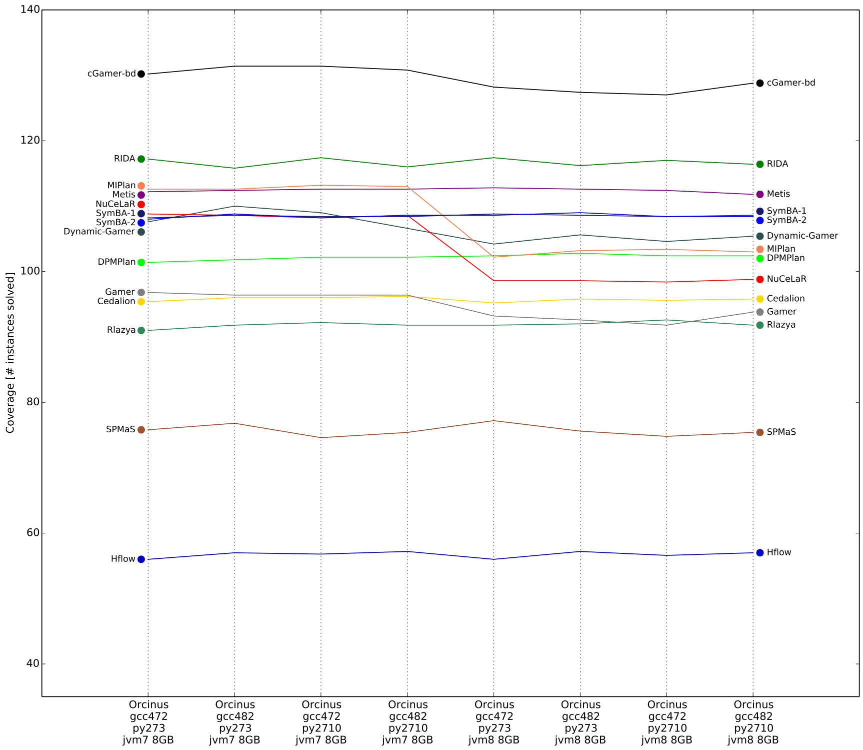


Fig. 9. Performance achieved by the planners of the *Optimal* track, in terms of number of instances solved, among different software configurations on *Orcinus*. The *Hpp* and *Hpp-ce* planners had a constant result of 15 problem instances solved and are not shown here.

completely to hardware or software environments different from those used in the competition. This may partially be due to the fact that many planners show similar (comparable) performance, but the impact of hardware and software environment can also dramatically change performance. In fact, for both of our considered IPC tracks we empirically observed a different top planner than in the official competition results. These hardware and software environment changes can be as minor as the version of the compiler used to create each solver executable: in our case GCC 4.7.2 vs. 4.8.2.

Furthermore, we also provide empirical evidence for the common belief that the choice of competition running time cutoffs and memory limits affect solver performance differently for different solvers, and thus can affect the resulting rankings.

While our experimental observations suggest that competition performance results should be carefully interpreted, we caution that these observations should not be taken as making past competition results somehow invalid, or diminishing the utility of solver competitions in general. Given our experimental results, we do recommend that users evaluate as many of the top-ranked solvers as possible in their own hardware and software environments when making decisions about the “best” solver for a specific problem.

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. Specifically for the IPC, increasing the memory limits used to 8 GB does appear to result in a general improvement of planners’ performance on existing benchmark instances, thus possibly providing a better snapshot of the actual performance of considered planners, and performing multiple planner runs on each benchmark instance would also help limit variance with only minimal additional human effort. Allowing competitors the ability to customise their own software configuration for the competition would potentially reduce this source of variation, but would also have a side effect of newly biasing the competition results toward competitors with the sophisticated knowledge, computational resources and time to do the performance tuning required.

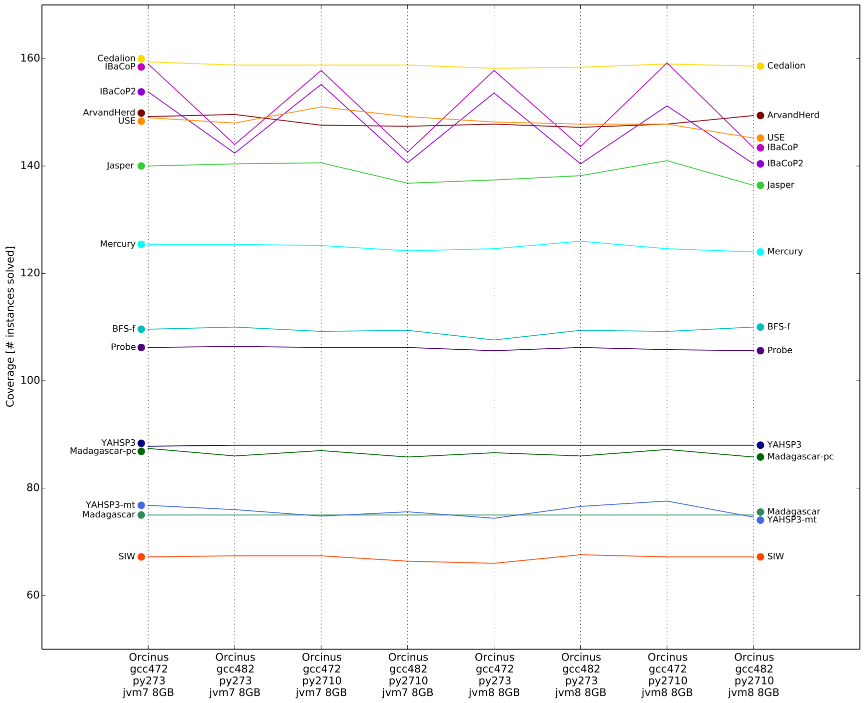


Fig. 10. Number of problem instances solved (instance coverage) by planners from the *Agile* track, using our 8 different software configurations on *Orcinus*.

We see several possible avenues for future work: first, a deeper investigation into specific planners such as *SymBA*, *SPMaS*, *MIPlan* and *DPMPlan*, which exhibited extremely large performance swings between our two hardware configurations; second, using the knowledge gained in this work, the study and development of a competition measuring solver performance across several distinct hardware and software environments; finally, a thorough analysis of additional sources of performance variation not covered in this paper, including benchmark instance set selection and solver stochasticity.

Acknowledgements

HH and CF gratefully acknowledge support through an NSERC Discovery Grant. This research was made possible by a resource allocation grant on the WestGrid Orcinus cluster by Compute-Calcul Canada to HH and computing time on the Galileo cluster allocated by CINECA to AG.

References

[1] Belov, A., Diepold, D., Heule, M.J. & Järvisalo, M. (2014). *SAT Competition 2014*.

[2] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, **115**(3), 211–252.

[3] Simon, L., Le Berre, D. & Hirsch, E.A. (2005). The SAT 2002 competition. *Annals of Mathematics and Artificial Intelligence*, **43**(1-4), 307–342.

[4] Gebser, M., Maratea, M. & Ricca, F. (2016). What's Hot in the Answer Set Programming Competition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (pp. 4327–4329).

[5] Stuckey, P.J., Feydy, T., Schutt, A., Tack, G. & Fischer, J. (2014). The MiniZinc challenge 2008–2013. *AI Magazine*, **35**(2), 55–60.

[6] McDermott, D.M. (2000). The 1998 AI planning systems competition. *AI magazine*, **21**(2), 35–55.

[7] Vallati, M., Chrpa, L., Grześ, M., McCluskey, T.L., Roberts, M. & Sanner, S. (2015). The 2014 International Planning Competition: Progress and Trends. *AI Magazine*, **36**(3), 90–98.

[8] Howe, A.E. & Dahlman, E. (2002). A Critical Assessment of Benchmark Comparison in Planning. *Journal of Artificial Intelligence Research*, **17**(1), 1–33.

[9] Long, D. & Fox, M. (2003). The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research*, **20**, 1–59.

Table 7

Number of problem instances solved (instance coverage) among different software configurations for the *Optimal* track on *Orcinus*.

	Instance coverage			
Planner	gcc 4.7.2 Python 2.7.3 JVM 1.7	gcc 4.8.2 Python 2.7.3 JVM 1.7	gcc 4.7.2 Python 2.7.10 JVM 1.7	gcc 4.8.2 Python 2.7.10 JVM 1.7
cGamer-bd	130.2	131.4	131.4	130.8
RIDA	117.2	115.8	117.4	116
Metis	112.2	112.4	112.6	112.6
SymBA-1	108.2	108.6	108.4	108.4
SymBA-2	108	108.8	108.2	108.6
MIPlan	112.6	112.6	113.2	113
Dynamic-Gamer	107.6	110	109	106.6
NuCeLaR	108.8	108.6	108.2	108.6
DPMPlan	101.4	101.8	102.2	102.2
Cedalion	95.4	96	96	96.2
Gamer	96.8	96.4	96.4	96.4
Rlazya	91	91.8	92.2	91.8
SPMaS	75.8	76.8	74.6	75.4
Hflow	56	57	56.8	57.2
Hpp	15	15	15	15
Hpp-ce	15	15	15	15
	gcc 4.7.2 Python 2.7.3 JVM 1.8	gcc 4.8.2 Python 2.7.3 JVM 1.8	gcc 4.7.2 Python 2.7.10 JVM 1.8	gcc 4.8.2 Python 2.7.10 JVM 1.8
cGamer-bd	128.2	127.4	127	128.8
RIDA	117.4	116.2	117	116.4
Metis	112.8	112.6	112.4	111.8
SymBA-1	108.8	108.6	108.4	108.6
SymBA-2	108.6	109	108.4	108.4
MIPlan	102.2	103.2	103.4	103
Dynamic-Gamer	104.2	105.6	104.6	105.4
NuCeLaR	98.6	98.6	98.4	98.8
DPMPlan	102.4	102.8	102.4	102.4
Cedalion	95.2	95.8	95.6	95.8
Gamer	93.2	92.6	91.8	93.8
Rlazya	91.8	92	92.6	91.8
SPMaS	77.2	75.6	74.8	75.4
Hflow	56	57.2	56.6	57
Hpp	15	15	15	15
Hpp-ce	15	15	15	15

[10] Hoffmann, J. & Edelkamp, S. (2005). The Deterministic Part of IPC-4: An Overview. *Journal of Artificial Intelligence Research*, **24**, 519–579.

[11] Gerevini, A., Haslum, P., Long, D., Saetti, A. & Dimopoulos, Y. (2009). Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, **173**(5–6), 619–668.

[12] López, C.L., Celorrio, S.J. & Olaya, Á.G. (2015). The deterministic part of the seventh International Planning Competition. *Artificial Intelligence*, **223**, 82–119.

[13] Riddle, P.J., Holte, R.C. & Barley, M.W. (2011). Does Representation Matter in the Planning Competition? In *Proceedings of the Ninth Symposium of Abstraction, Reformulation and Approximation (SARA)* (pp. 90–98).

[14] Vallati, M. & Vaquero, T. (2015). Towards a Protocol for Benchmark Selection in IPC. In *Proceedings of the 4th Workshop on the International Planning Competition (WIPC)*.

[15] Hurley, B. & O'Sullivan, B. (2015). Statistical Regimes and Runtime Prediction. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 318–324).

[16] Ghallab, M., Nau, D. & Traverso, P. (2004). *Automated Planning: Theory and Practice*. Elsevier Science.

[17] Coles, A.J., Coles, A., Olaya, A.G., Celorrio, S.J., López, C.L., Sanner, S. & Yoon, S. (2012). A Survey of the Seventh International Planning Competition. *AI Magazine*, **33**(1), 83–88.

[18] Vallati, M., Chrpa, L. & McCluskey, T. (2017). What You Always Wanted to Know about the Deterministic Part of IPC 2014 (But Were too Afraid to Ask). *The Knowledge Engineering Review*.

[19] Vallati, M., Chrpa, L. & McCluskey, T.M. (2014). *The Eighth International Planning Competition. Description of Participant Planners of the Deterministic Track*.

[20] Coles, A., Fox, M. & Smith, A. (2007). A New Local-Search Algorithm for Forward-Chaining Planning. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS* (pp. 89–96).

[21] Cohen, E. & Beck, J.C. (2017). Problem Difficulty and the Phase Transition in Heuristic Search. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (pp. 780–786).

[22] Hoos, H.H. & Stützle, T. (1999). Towards a Characterisation of the Behaviour of Stochastic Local Search Algorithms for SAT. *Artif. Intell.*, **112**(1-2), 213–232.

[23] Kroc, L., Sabharwal, A. & Selman, B. (2010). An Empirical Study of Optimal Noise and Runtime Distributions in Local Search. In *Theory and Applications of Satisfiability Testing - SAT 2010, 13th International Conference, SAT* (pp. 346–351).

[24] Rossi, F., Van Beek, P. & Walsh, T. (2006). *Handbook of constraint programming*. Elsevier.

[25] Cheeseman, P., Kanefsky, B. & Taylor, W. (1991). Where the really hard problems are. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 331–337).

[26] Bylander, T. (1996). A probabilistic analysis of prepositional STRIPS planning. *Artificial Intelligence*, **81**(1–2), 241–271.

[27] Rintanen, J. (2004). Phase Transitions in Classical Planning: An Experimental Study. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS)* (pp. 101–110).

[28] Hoffmann, J. (2011). Analyzing search topology without running any search: On the connection between causal graphs and h+. *Journal of Artificial Intelligence Research*, **41**, 155–229.

[29] Hutter, F., López-Ibáñez, M., Fawcett, C., Lindauer, M., Hoos, H.H., Leyton-Brown, K. & Stützle, T. (2014). AClib: A benchmark library for algorithm configuration. In *Proceedings of the Eighth International Conference on Learning and Intelligent Optimization (LION)* (pp. 36–40).

[30] Helmert, M., Röger, G. & Karpas, E. (2011). Fast Downward Stone Soup: A baseline for building planner portfolios. In *Proceedings of the Third Workshop on Planning and Learning (ICAPS-PAL)* (pp. 28–35).

Table 8

Performance, in terms of coverage and IPC runtime score, of planners of the *Agile* track run on different software architectures, on the *Orcinus* cluster.

	gcc 4.7.2		gcc 4.8.2		gcc 4.7.2		gcc 4.8.2	
Planner	Python 2.7.3		Python 2.7.3		Python 2.7.10		Python 2.7.10	
	JVM 1.7		JVM 1.7		JVM 1.7		JVM 1.7	
	IPC score	coverage	IPC score	coverage	IPC score	coverage	IPC score	coverage
Cedalion	107.4	159.4	106.2	158.8	106.9	158.8	107.6	158.8
IBaCoP	70.1	159	65.8	144	70	157.8	64.9	142.6
USE	79.3	149	77.7	148	79.6	151	78.1	149.2
ArvandHerd	89.2	149.2	90.2	149.6	88.1	147.6	88.2	147.4
IBaCoP2	59.6	153.8	56.6	142.4	60	155.2	55.7	140.6
Jasper	84.8	140	83.7	140.4	84.3	140.6	79.9	136.8
Mercury	67.8	125.4	67.5	125.4	67.8	125.2	66.5	124.2
BFS-f	66.1	109.6	66.4	110	66.2	109.2	66.7	109.4
Probe	71.4	106.2	71.2	106.4	71.3	106.2	71.4	106.2
YAHSP3	73.7	87.8	73.6	88	73.5	88	73.2	88
Madagascar-pc	63.4	87.4	62.6	86	63.4	87	62.6	85.8
YAHSP3-mt	56	76.8	54.2	76	54.2	74.8	53.8	75.6
Madagascar	64.1	75	63.9	75	64	75	63.9	75
SIW	49.6	67.2	50.2	67.4	50.1	67.4	49.8	66.4

	gcc 4.7.2		gcc 4.8.2		gcc 4.7.2		gcc 4.8.2	
	Python 2.7.3		Python 2.7.3		Python 2.7.10		Python 2.7.10	
	JVM 1.8		JVM 1.8		JVM 1.8		JVM 1.8	
	IPC score	coverage	IPC score	coverage	IPC score	coverage	IPC score	coverage
Cedalion	105.8	158.2	107.1	158.4	107.6	159	107	158.6
IBaCoP	69.1	157.8	66	143.6	69.8	159.2	66.2	143.4
USE	78.1	148.2	78.1	147.8	77.6	147.8	75.7	145.2
ArvandHerd	87.9	147.8	87.5	147.2	88.8	147.8	89.8	149.4
IBaCoP2	58.1	153.6	55.6	140.4	58.6	151.2	56.1	140.4
Jasper	82.3	137.4	82.3	138.2	84.5	141	79.7	136.4
Mercury	66.4	124.6	67.7	126	67.1	124.6	67	124
BFS-f	64.7	107.6	66.1	109.4	66.1	109.2	66.5	110
Probe	70.7	105.6	71.4	106.2	71.5	105.8	71.2	105.6
YAHSP3	73.3	88	73.1	88	73.8	88	73.5	88
Madagascar-pc	63.1	86.6	62.6	86	63.2	87.2	62.4	85.8
YAHSP3-mt	53.7	74.4	54.4	76.6	56	77.6	53.4	74.6
Madagascar	63.9	75	64.1	75	64	75	63.3	75
SIW	49.4	66	50.3	67.6	50	67.2	50.1	67.2