Artificial Intelligence 223 (2015) 82–119

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

The deterministic part of the seventh International Planning Competition[☆]

Carlos Linares López[*], Sergio Jiménez Celorrio, Ángel García Olaya

Computer Science Department, Universidad Carlos III de Madrid, Leganés (Madrid), Spain

ARTICLE INFO

Article history:
Received 22 November 2013
Received in revised form 12 December 2014
Accepted 19 January 2015
Available online 3 February 2015

Keywords:
Automated planning
Planning systems
International Planning Competition
Benchmarks for planning
Experimental evaluation of planning systems

ABSTRACT

The International Planning Competition is organized in the context of the International Conference on Automated Planning and Scheduling (ICAPS) and it is considered a reference source for the planning and scheduling community. The competition is typically organized every two years and deals with relevant issues for the community such as the definition of evaluation standards, the publication of benchmarks and the collection and dissemination of data about state-of-the-art planners. This paper focuses on the deterministic part, the longest-running part of the International Planning Competition. The paper describes its format, the participants, the selection of benchmarks and the generated results accompanied with analysis from different perspectives. The paper also examines the results of a brand new track created to explore the potential of planners that exploit the power of multi-core processors. Overall, the results of the competition indicate significant progress with respect to previous competitions, but they also reveal that some issues remain open and need further research, such as the coverage of temporal planners when concurrency is required and the performance in the multi-core track. As a novelty, all the data and the software generated for running the competition have been made publicly available allowing researchers to reproduce the competition and to carry out different analysis of the results.

1. Introduction

The International Planning Competition (IPC) has typically been held every 2 years in the context of the International Conference on Automated Planning and Scheduling (ICAPS). The main activity in the competition is evaluation of state-of-the-art planners by running them on a set of planning problems and comparing their performance with some specific metrics. Although the IPC is competitive and awards planners for their performance, the main goals of the competition are collection and dissemination of data and definition of evaluation methodologies. In fact, the IPC is used as a reference source when building a planner, and most new planning techniques are evaluated by considering languages, benchmarks, and metrics defined in the IPC series.

Currently the IPC comprises three parts: (1) a deterministic part for evaluation of domain-independent planners in deterministic and fully observable environments; (2) a learning part for planners able to learn and exploit domain-specific knowledge in deterministic planning; and (3) an uncertainty part for domain-independent planners able to plan under un-

[☆] This paper was submitted to the Competition Section of the journal.
[*] Corresponding author.
E-mail addresses: carlos.linares@uc3m.es (C. Linares López), sjimenez@inf.uc3m.es (S. Jiménez Celorrio), agolaya@inf.uc3m.es (Á. García Olaya).

http://dx.doi.org/10.1016/j.artint.2015.01.004
0004-3702/© 2015 Elsevier B.V. All rights reserved.

certainty. The deterministic part has been the main focus of the IPC over the years and is the part addressed in this paper. Details on the other two parts can be found in a survey paper [1] and on the IPC website.[1]

After a 3-year gap, the seventh edition of the deterministic part intentionally provided continuity with the previous competition, IPC-2008, for easier comparisons and to build iteratively on previous work that could benefit the community. For example, IPC-2011 was structured using the same tracks as for IPC-2008 and preserved the same input language and evaluation scores. Likewise, the competition reused domains and problems to better quantify progress in the field. However, IPC-2011 also introduced several new planning domains and a large collection of problems over these domains that can serve as a reference for future research. There was special emphasis on the temporal track, and planning problems that require concurrency (i.e., that at least two different actions have to be performed at the same time) were intentionally included. With the aim of exploring new directions for planning research, the seventh edition introduced a multi-core track that evaluates the performance of planners with algorithms able to exploit multi-processor machines.

There were also similarities and differences with regard to the software used to run the competition. Continuing the work achieved in IPC-2008, there were explicit efforts to highlight the transparency and reproducibility of the results of IPC-2011. This involved the creation of a public repository of all the software, benchmarks, and source codes for the participant planners, along with short papers describing them and all the data generated. This public resource allows researchers to validate the competition results using their own means for further analysis from different perspectives.

The seventh edition of the deterministic part attracted a record number of 55 participants grouped in 31 teams from 11 different countries: Australia, Canada, China, France, Germany, India, Israel, Italy, Spain, UK, and USA. This is almost eight times the number of participants in the first competition, IPC-1998, and double the number in the previous one, IPC-2008. The competition results were presented at a special ICAPS session in July 2011. After the competition, a more detailed analysis was performed to gain more insight into the results and possible causes and consequences. In summary, the main contributions of the deterministic part of IPC-2011 are:

- A collection of new domains and their corresponding problem generators.
- A detailed evaluation of the relative performance of the IPC-2011 planners with respect to new and previously used benchmarks.
- A new track for evaluation of multi-core planners.
- A public repository containing all the data produced during the competition and open-source tools for running the experiments and analyzing the results.
- A detailed analysis of the competition results that provides insights into the performance of state-of-the-art planners from different perspectives such as coverage, quality, CPU time, and memory usage.

The overall aim of this paper is to provide accurate answers to the following questions:

- *What is being measured?* The definition of the scoring schema and how it is affected by the particular goal of every track is discussed in Section 2.3. The characterization of every track is described in Section 2.1. The scoring schema adopted is critically analyzed in Section 6 and various alternatives are examined.
- *What are the benchmarks?* Selection of the benchmark suite is a difficult problem that is addressed in Section 3. The difficulty mainly arises from the need to select challenging yet solvable problems and the fact that it might introduce bias, as discussed in Section 3.2 and Appendix C.1.
- *What are the results?* The main results are presented in Section 4 and are compared with the results from IPC-2008 and closely related tracks in Section 5.
- *General questions and trends* are summarized in Section 7.

The remainder of the paper is organized as follows. Section 2 describes the deterministic part of IPC-2011 in terms of its format, the participants, and the evaluation schema. Section 3 reviews the benchmarks used and details the mechanisms followed in the domain and problem selection. Section 4 analyzes the competition results with respect to coverage, quality, raw speed, and memory usage. Section 5 presents a scalability analysis. The performance of the top planners in IPC-2011 is compared with that of the top planners in IPC-2008, and the performance of parallel solvers is compared with their sequential counterparts. The scoring schema used is critically analyzed in Section 6 and a number of alternatives are discussed. Section 7 summarizes with a number of conclusions. A series of appendices are available online as Supplementary material for further reference. Appendix A describes the participants for all the tracks. Appendix B provides additional details about the selected domains. Appendix C presents results for additional experiments performed once the competition was over. Appendix D shows a novel approach used to select problems that were reused from previous IPCs. Appendix E provides additional details on the way in which various statistical tests were conducted.

[1] http://ipc.icaps-conference.org/.

2. The deterministic part of the seventh International Planning Competition

One of the main aims in designing the deterministic part of IPC-2011 was to make the competition as inclusive as possible. The benchmarks were set up to have low PDDL requirements, mostly STRIPS plus simple use of numeric fluents. This fact, in conjunction with the public availability of the source code for the FAST DOWNWARD planning system,[2] made the deterministic part of IPC-2011 extremely popular with a record number of entrants.

2.1. Format

The deterministic part of the IPC currently comprises three different planning models that differ in expressiveness. In all cases the evaluation of planners is carried out by comparing their performance with respect to the quality of the plans found. There are, however, different definitions for quality according to the purpose of each model.

Sequential planning pursues the generation of a sequence of actions in deterministic and fully observable environments. In sequential planning, quality is evaluated inversely to the total cost of the solution, which is defined as the sum of the individual costs for all actions included in the solution. Thus, maximization of quality corresponds to minimization of the total cost of the solution plans. *Planning with preferences* involves the generation of plans when goals have different strength requirements, for example, motivated by soft goals or trajectory constraints [2]. The objective of planners in this task is to maximize the total net benefit, that is, the difference in penalties for not achieving some goals or not observing a number of constraints and the total action cost. Finally, *temporal planning* involves the generation of solution plans when actions have a duration and may temporarily overlap. In this case, plan quality is inversely related to the makespan of the plan, that is, the temporal duration from when the first action is initiated to when the last action is completed.

Similar to the 2011 SAT competition [3], IPC-2011 explicitly distinguished between the ability to use resources as efficiently as possible and the ability to provide fast solutions when using all available resources. The first case corresponds to the case of *sequential* solvers, whereas the second is explicitly designed for *parallel* solvers. Traditionally, all tracks of the IPC series were sequential, but in 2011 a brand new track was created for evaluation of parallel solvers: the multi-core track. In this track, all planners were given 30 min of wall-clock time regardless of the number of processes and/or threads launched, whereas in the sequential tracks, planners were given 30 min of CPU time in total. Nevertheless, parallel solvers were allowed to enter sequential tracks since CPU time was accumulated across all processes and threads launched by the solver, which resulted in reduced real times for parallel solvers.

Each planner was run on a single node of the cluster and no planner was allowed to use more than a single dual core, except in the multi-core track, in which four dual cores were readily available to speed up the processing. In all tracks, each planner was allotted 30 min per planning task (either CPU time or wall-clock time) and a memory limit of 6 GB of RAM and 750 GB of hard disk space.

All competitors were required to adhere to the syntax of VAL [4], which was used to retrieve various statistics (e.g., the total cost of each solution, the number of valid solutions, the step length of the plan) and to facilitate automated validation. In this regard, two criteria were set:

1. If a planner generated any invalid solution, it was disqualified for that particular problem even if other valid solutions were generated.
2. For the sequential optimal track, if a planner generated any suboptimal solution, it was assigned a null score for the whole domain.

2.2. Participants

The deterministic part of IPC-2011 received 55 submissions distributed among four different tracks. Table 1 summarizes the number of planners covering different PDDL fragments for each track. Different entries are counted separately even if they implemented different versions of the same planner (e.g., FDSS-1 and FDSS-2).

Although PDDL3.1 was the input language for IPC-2011, none of the 55 entrants fully implemented it. The vast majority of participants only supported the STRIPS subset besides typing, simple numeric fluents, and the equality predicate. Less than half of the planners implemented the full PDDL1.2 set, including the ADL requirements. None of the planners implemented PDDL2.2 in full, with only 14 supporting derived predicates and only three supporting timed-initial literals. Finally, only three planners from the temporal satisficing track implemented some limited support of PDDL3.1, in particular numeric state variables.

Next we give an overview of participation in the different competition tracks. Fine-grained details of the 55 participant planners can be found in Appendix A. For further details, interested readers are referred to the short papers submitted by the authors of each planner, which are available as a technical report [5].

[2] http://www.fast-downward.org/.

Table 1
PDDL coverage of the competing planners and total number of entrants per track. A dash indicates that a feature is not applicable in a particular track.

Feature	PDDL	Satisficing	Optimal	Multi-core	Temporal
Typed representations	1.2	27	12	8	8
Untyped representations	1.2	21	12	7	7
Schematic representations	1.2	27	12	7	8
Grounded representations	1.2	23	1	7	6
Negative conditions	1.2	16	1	6	0
ADL conditions	1.2	15	1	6	1
Conditional effects	1.2	15	0	5	1
Universal effects	1.2	18	1	5	2
Derived predicates	2.2	11	0	3	0
Time-initial literals	2.2	–	–	–	3
Numeric state variables	3.1	–	–	–	3
Object fluent representations	3.1	0	0	0	0
Total		27	12	8	8

Sequential track. The sequential track comprises an optimal track, in which planners must provide the best solutions in terms of the total action cost, and a satisficing track, in which planners can provide suboptimal solutions.

For the sequential optimal track, 22 planners were registered and 14 were finally submitted, of which two were later withdrawn, leaving 12 entrants. Ten out of the 12 participant planers were built on top of FAST DOWNWARD, with pure heuristic search being the most popular approach, followed by 8 out of 12 planners. The main difference among these planners was in the search algorithms they implemented (A^* and LM-A^* used by 5 and 3 planners, respectively) and the admissible heuristic functions they used to guide the search ($h^{\max}$, LM-cut, Merge and Shrink, ...).

For the sequential satisficing track, 29 planners were registered, of which 27 were submitted. Many of the planners submitted for this track implement anytime strategies that allow improvements in the quality of the first plan found until the whole state space is exhausted or the planning is terminated after the allotted time or memory is exhausted.

Multi-core track. Current computers with multiple cores are now affordable and some researchers have already started to work on multi-core planning [6–8], so IPC-2011 introduced a new track to evaluate their potential. Unfortunately, it was not possible to use a graphics processing unit (GPU), which has shown promising performance [9]. The aim of this track was to evaluate the performance of planners using multiple cores at the same time (in contrast to a distributed environment, in which different computers can be used simultaneously) and observe whether they can outperform standard planners that traditionally use only one processor.

Only a sequential satisficing track was arranged for the multi-core track.[3] Ten planners were registered, of which eight were submitted. Two of these, ACOPLAN and ROAMER-P, were parallel versions of entries submitted to the sequential satisficing track. Among the other six, MADAGASCAR and MADAGASCAR-P were exactly the same planners as those submitted to the sequential satisficing track, and two planners exploited the multi-core setting by running a number of independent searches in parallel. Specifically, one planner implements dove-tailing (simultaneous runs of the same algorithm with different parameter settings) on a frontier search procedure. The other planner simultaneously runs different members of an algorithm portfolio. The remaining two planners parallelize widely used planning algorithms, such as enforced hill-climbing and best-first searches, by implementing parallel procedures for evaluating and expanding nodes.

Temporal track. In this track, planners are required to find a valid solution to a planning task that involves durative actions that might temporarily overlap or interfere. It has already been noted that temporal planning can be computationally more complex than classical planning [10] and thus a separate track was arranged for this context.

Twelve planners were registered but only eight were submitted. Evaluation of this track in IPC-2011 focused on a conservative fragment of temporal planning according to PDDL2.1 semantics that included durative actions with and without a concurrency requirement. Only two planners (POPF2 and LMTD) provided concurrency support.

Preferences track. As in IPC-2008, a preferences track for IPC-2011 was announced and later canceled because four planners were registered for the satisficing track but only one was submitted. Likewise, six planners were registered for the optimal track but only one was submitted.

This may be not a coincidence since effective compilations for planning with preferences have recently appeared that transform some preference planning tasks into classical planning tasks [11], with better results than those of planners tailored for this track, at least in the optimal setting. Participation in the preferences track requires handling of complex parts of PDDL (e.g., full support for fluents or disjunctive goals). In addition, many of

[3] From the start, this track was devoted to non-optimal planning and therefore the term *sequential satisficing multi-core* might be more appropriate. However, since there was no sequential optimal multi-core track, the term *satisficing* is omitted here.

the participants who registered for this track were also participating in the sequential tracks, so they may have eventually decided to concentrate on the latter.

2.3. Evaluation

Since its conception, the IPC has aimed to provide the planning community with standard mechanisms that objectively and reliably evaluate the performance of different planning techniques for easier comparability. Throughout the IPC series, different evaluation schemes have been proposed to score and rank planners and after seven editions this ideal has proved to be a difficult challenge. The difficulty lies in the nature of planning, which lends itself to a number of very diverse and often conflicting evaluation criteria from straightforward binary success measures (such as finding a solution or proving that there is none) to optimization along many different dimensions. Furthermore, the evaluation has to measure the domain independence of planners, quantifying their versatility for different classes of problems across different domains. Therefore, planner performance can be analyzed from different perspectives, such as the number of problems solved, CPU time, plan length, total cost, makespan, or other features such as the diversity or flexibility of the solutions. For this reason, selection of the best planner necessarily depends on the context in which it will be used.

An important consideration in IPC-2008 was that quality, defined inversely to the total cost of a plan, is widely regarded as a useful measure in most contexts. From a theoretical point of view, finding a solution is not the only interesting challenge, and improving over previous solutions has received much attention in the literature [12,13]. From a practical point of view, although faster identification of solutions is a primary concern in some real-world applications, finding good solutions is also relevant, especially if the cost of the plans is directly related to important features such as time or economic costs so that large time horizons are allowed for planning. The deterministic part of IPC-2011 continued with the same evaluation scheme defined for the previous edition. This focuses on good plan quality, with less emphasis on the number of problems solved or the CPU time used. Accordingly, if two planners find solutions within the same time and memory bounds, the plan with the lower cost is awarded a better score. This is not applicable to the optimization track, for which all solutions are expected to have the same cost and thus only the number of solutions found (or *coverage*) is relevant for the final score.

For IPC-2011, all planners were given the same time and memory cutoffs (1800 s and 6 GB of main memory) and were required to find valid solutions with the best *quality*. For a particular planner with regard to a given planning task, quality is defined as the ratio of the lowest total cost (computed as the sum of the costs of all individual actions included in a solution plan) to the total cost of the best solution found. Thus, each planner p gets a score per planning task i, S_i^p, expressed as

$$S_i^p = \frac{C_i^*}{C_i^p},$$

where C_i^p is the total cost of the best solution found by planner p for instance i, and C_i^* is the lowest total cost found so far by any planner for the same problem, that is, $C_i^* = \min_p\{C_i^p\}$. The final score for each planner, S^p, is computed as the sum of the scores obtained in every planning instance, aggregating scores among domains of the same track, $S^p = \sum_i S_i^p$. Obviously, scores were not aggregated among tracks since they were considered as separate competitions.

No optimal solver was built for any domain and thus all the scores computed in this way rely solely on the total cost of the plans found by the different competitors. Although the final scores might change in the light of the true optimal scores (even resulting in different rankings for planners), the results and conclusions drawn here are expected to be robust. Section 6 discusses the current scoring schema and analyzes alternatives.

Although the score is computed considering only the quality of solutions, coverage and CPU time are arguably implicit to some extent. On the one hand, unsolved problems are scored as 0, so that coverage is acknowledged by the score function. On the other hand, all planners were given 30 min per planning task, so that planners that quickly identify solutions can dedicate more time to improving the quality of the initially produced plans. Next we discuss how the scoring function behaves in every track.

Sequential track. In the case of the sequential optimizing track, the score function equals coverage, or the number of problems solved: each planner is awarded one point for every problem solved, and zero otherwise.

In the case of the satisficing track, the score function maps the quality of every planner in the range $[0, 1]$ inversely to the total cost of its solutions, using the best known solution as a reference.

Multi-core track. Planners in this track were scored using the same schema as for the sequential satisficing track.

Temporal track. In this track there is no notion of total cost, although costs can of course be defined in general for temporal planning. Instead, quality is defined inversely to the makespan of every solution: $S_i^p = \frac{M_i^*}{M_i^p}$, where M_i^p denotes the makespan of the best solution found by planner p for instance i, and M_i^* is the shortest makespan found so far by any temporal planner for the same problem.

The first and second best-ranked planners were picked as the winner and runner-up, respectively, although additional considerations were scored according to a judgment call, in particular in the sequential satisficing and temporal satisficing tracks (Sections 4.3.4 and 4.5.4).

3. Benchmarks

Researchers are increasingly evaluating their results with regard to the benchmark selection of the IPC series. Therefore, this selection is an important issue not only for the competition itself but also for the planning community. Ideally, the IPC benchmarks should cover different ranges for the structure and difficulty of problems. However, guaranteeing this coverage is complex because of the lack of domain-independent techniques for assessing the diversity and difficulty of planning problems.

Similar problem-solving competitions, such as the SAT competition, address this issue for random track-generating benchmarks using random synthetic problems of varying difficulty. The phase transition of such SAT instances is well known [14] and progress is then often measured in this particular track in terms of the size (number of variables and clauses) of the formulas that can be tackled. Unfortunately, this is not true for other tracks of the same competition, such as the structured/application tracks. Similarly, the case of automated planning is complex because the branching factor and the depth of the planning problems are not easily bounded with the number of state variables. Although a phase transition has already been demonstrated in automated planning [15–17], it is relative only to random graphs that have never been used in previous IPCs. In general, not much is known about what makes some problems particularly harder than others, although some research has been carried out [18,19].[4]

In other competitions, such as the Answer Set Programming competition, the organizers aim to provide a balance between problem categories such as *search*, *query*, and *optimization* covering different computational complexity classes such as *polynomial*, *NP*, and *Beyond NP* [20]. This classification is very difficult in automated planning, for which a better known result is that classical planning is PSPACE-complete [21] (even in very restricted cases), so that the complexity analysis falls in a case-by-case analysis [22,23]. Other studies have analyzed the structure of planning domains regarding a particular search configuration with promising results, such as creating taxonomies of domain classes under the heuristic function h^+ [24,25] or others [26]. Finally, in other cases, such as the CSP competition, the organizers only select problems that can be solved by at least one of the participants in a sensible time frame, and a somewhat related approach was followed in IPC-2011 (Section 3.2 and Appendix D).

Another consideration when building the IPC benchmarks is the inclusion of real-world problems [18]. The difficulty here arises from the shortage of applications that use standard PDDL.

With all this in mind, the final design of the IPC-2011 benchmarks was motivated by three goals: (i) to evaluate the domain independence of planners; (ii) to assess the evolution of planners since IPC-2008; and (iii) to provide interesting domains for the planning community. In accordance with these objectives, there was special emphasis on evaluation of planners over a large number of different domains instead of over a large number of planning tasks from a reduced number of domains. In addition, a good number of domains and problems were reused from the deterministic and learning tracks of IPC-2008, but new domains were also included in response to a public call for domains. Finally, challenging test sets for state-of-the-art planners of increasing difficulty were designed. The remainder of this section surveys the domains and problems selected. Further details on the particular domains and problems generated for each domain are in Appendix B.

3.1. Selection of domains

The benchmarks of IPC-2011 comprised 19 different domains: 14 domains in the sequential optimal, satisficing, and multi-core tracks, and 12 domains in the temporal satisficing track. Eleven domains were reused from previous competitions, and eight new domains were introduced for the first time. Among the new domains, six were collected in response to a public call for domains, and two, BARMAN and TURNANDOPEN, were created by the organizers. In contrast to the practice in previous competitions, there was a single PDDL definition per domain because most of the planners submitted only supported basic PDDL requirements (Table 1, page 85). The IPC-2011 organizers believed that efforts to make different versions would not be worthwhile because few planners would have benefited.

Tables 2 and 3 show the domains used in IPC-2011 for the sequential and temporal tracks, respectively, along with their PDDL requirements. Domains shown in bold font were used in all tracks. All the domains use at most STRIPS plus actions costs and action durations in the temporal track. In the sequential tracks, eight domains were reused from the deterministic track of IPC-2008. Another domain (PARKING) was from the learning track of IPC-2008 and five domains were new. Four of the 14 domains are inspired by real applications (ELEVATORS, PARCPRINTER, SCANALYZER and TRANSPORT), while the remaining 10 explore different structures of planning problems.

The new domains were especially created to challenge specific planning techniques. In the sequential track we introduced NOMYSTERY, BARMAN, PARKING, FLOORTILE and VISITALL to challenge diverse weaknesses of the delete-relaxation heuristics. The management of limited resources is a key issue in automated planning however the delete-relaxation ignores resource consumption producing optimistic estimations of limited benefit in resource-constrained problems. More precisely delete effects in the NOMYSTERY domain encode the fuel consumptions of shippings; they encode the fact that robot hands can

[4] This is not to say, however, that random graphs cannot be used in the IPC; indeed, it might be a good idea to do so, although none was used in IPC-2011.

Table 2
Domains used in IPC-2011 for the sequential optimal, satisficing, and multi-core tracks. Domains shown in bold font were also used in the temporal satisficing track.

Domain name	Origin	PDDL requirements
BARMAN	New	`:typing :action-costs`
ELEVATORS	IPC-2008	`:typing :action-costs`
FLOORTILE	New	`:typing :action-costs`
NOMYSTERY	New	`:typing :action-costs`
OPENSTACKS	IPC-2006	`:typing :action-costs`
PARCPRINTER	IPC-2008	`:typing :action-costs`
PARKING	IPC-2008 (learning)	`:typing :action-costs`
PEGSOL	IPC-2008	`:typing :action-costs`
SCANALYZER	IPC-2008	`:typing :action-costs`
SOKOBAN	IPC-2008	`:typing :action-costs`
TIDYBOT	New	`:typing :equality`
TRANSPORT	IPC-2008	`:typing :action-costs`
VISITALL	New	`:typing`
WOODWORKING	IPC-2008	`:typing :action-costs`

Table 3
Domains used in IPC-2011 for the temporal satisficing track. Domains shown in bold font were also used in the sequential tracks.

Domain name	Origin	PDDL requirements
CREWPLANNING	IPC-2008	`:typing :durative-actions`
ELEVATORS	IPC-2008	`:typing :durative-actions`
FLOORTILE	New	`:typing :durative-actions`
MATCHCELLAR	New	`:typing :durative-actions`
OPENSTACKS	IPC-2006	`:typing :durative-actions`
PARCPRINTER	IPC-2008	`:typing :durative-actions`
PARKING	IPC-2008 (learning)	`:typing :durative-actions`
PEGSOL	IPC-2008	`:typing :durative-actions`
SOKOBAN	IPC-2008	`:typing :durative-actions`
STORAGE	IPC-2006	`:typing :durative-actions`
TMS	New	`:typing :durative-actions`
TURNANDOPEN	New	`:typing :durative-actions`

only grasp one object at a time and that glasses need to be clean before being filled in the BARMAN domain and, in the PARKING domain, delete effects encode when curbs are available for parking. In addition the FLOORTILE domain was introduced because its dead-ends are difficult to recognize with a delete-relaxation heuristic and the VISITALL domain was introduced because of its conflicting goals where progressing towards one goal means moving away from the others so that large plateaux are produced when using delete-relaxation heuristics.

The TIDYBOT domain was introduced with a different motivation, the increasing interest in re-approaching the fields of AI planning and autonomous robotics. State-of-the art planners fail to address problems with large state spaces, like the motion planning problems typically addressed in robotics. The TIDYBOT domain models a household cleaning task that involves motion planning and task planning in which one or more robots must pick up a set of objects and put them into goal locations.

In the temporal track there were four new domains and eight domains from previous IPCs (six from IPC-2008, one from IPC-2006, and one from the learning track of IPC-2008), although some of them had to be adapted to the requirements supported by the competing planners. In this case, four domains are also inspired by real-world applications (CREWPLANNING, ELEVATORS, PARCPRINTER, and TMS). The IPC-2008 domains MODELTRAIN, TRANSPORT, and WOODWORKING were not reused as they require `:numeric-fluents`, which is not supported by five out of the eight temporal planners.

Most temporal domains in previous competitions were temporally simple, in the sense that they did not require concurrency and problems could be solved by a pure sequential planner [27].[5] The IPC-2011 benchmarks intentionally included three new domains, MATCHCELLAR, TMS, and TURNANDOPEN, in which all possible solutions require concurrency of actions. In particular, in the MATCHCELLAR domain, a set of fuses have to be mended in the light of a match. In the TURNANDOPEN domain a robot must turn the doorknob and push the door at the same time to navigate through rooms and in the TMS domain diverse pieces of ceramic have to be baked and treated while a kiln is on fire. These domains present a greater challenge to temporal planners, and the organizers recognized this by giving a *runner-up* award to the planner with the best performance in these domains (Section 4.5.4).

[5] Notable exceptions are the *time-timewindows-compiled* versions of the domains Satellite, UMTS, and Pipesworld–NoTankage introduced in the fourth IPC.

Other people contributed with additional domains that were not included in the final version. The main reason for discarding them was that they required fragments of PDDL that only a few competing planners (if any) were able to support. Although these domains were not used in the final version of the competition, they are available on the competition website[6] with a detailed description and the specific reasons why they were discarded.

3.2. Selection of problems

The IPC-2011 benchmarks consists of 20 problems per domain. This number is smaller than in previous competitions, but the total number of problems is compensated by including a larger number of domains. For example, IPC-2008 had six, eight, or nine domains per track with 30 problems each, resulting in an overall number of planning tasks ranging from 180 to 270. However, IPC-2011 had 20 planning tasks in 12 or 14 different domains per track, resulting in an overall number of instances ranging from 240 to 280. In all, the IPC-2011 benchmarks comprised 800 problems distributed as follows: 280 problems for the sequential optimal track, 280 for the sequential satisficing and multi-core tracks, and 240 for the temporal satisficing track.

From a general point of view, every track poses different challenges that should be taken into account. For example, although there is no difference in theoretical complexity in the general case, optimal planning is harder than satisficing sequential planning in practice [22,28]. Therefore, planners in both tracks were evaluated over the same collection of domains, but problems for the optimal track were carefully selected to be easier to solve. In addition, to facilitate direct comparisons, planners in the multi-core track were faced with exactly the same planning tasks used in the sequential satisficing track (Section 5.2).

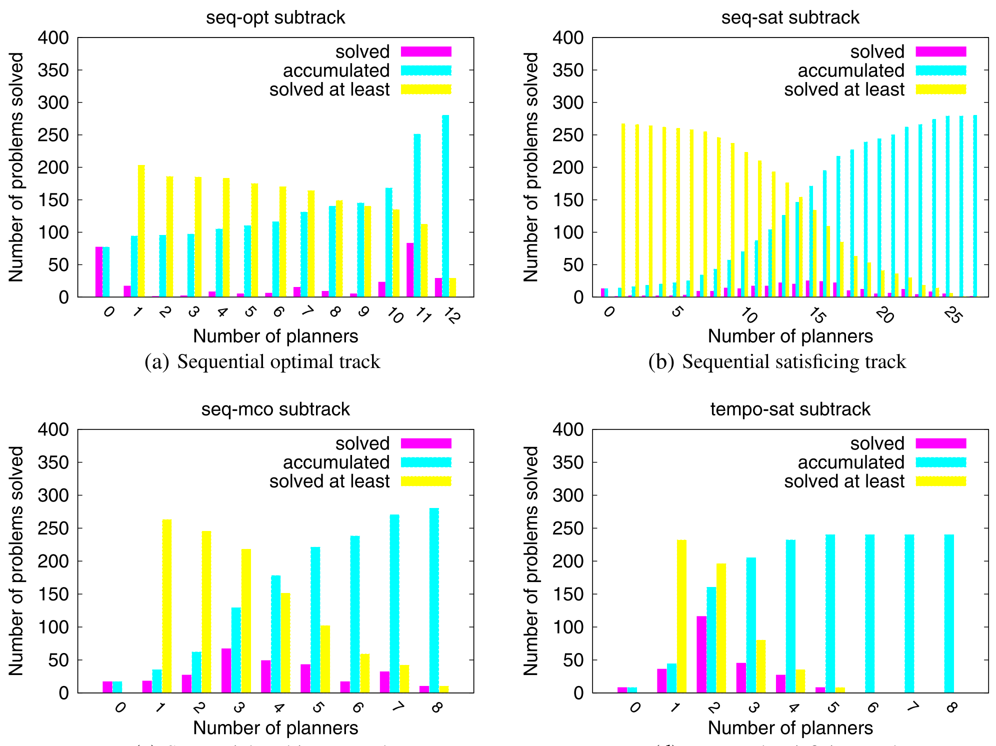
In each track, problems with increasing difficulty were selected so that they would be challenging (but not impossible) for state-of-the-art planners. This criterion requires identification of the transition between solvable and non-solvable problems (from a practical point of view and not whether they can be actually solved or not) for each domain and track for current state-of-the-art planners. Although the number of objects in a planning problem has been widely used as a weak measure of difficulty, there is currently no effective domain-independent procedure for characterizing the difficulty of problems for a given set of planners, other than attempting to solve them and to look at the results, as in the CSP competition. This approach would require a preliminary IPC to select problems for the official IPC, which, given the large number of participants, was intractable.

Therefore, two different methods were adopted for selection of problems, depending on whether data on their difficulty were available (e.g., when using problems from previous IPCs). Arguably, both approaches assume that the transition from solvable to non-solvable for IPC-2011 competitors would be close to the transition experienced by a different set of planners.

For the new domains introduced in IPC-2011 we developed (or asked domain creators to develop) a problem generator for each domain. The parameters of these generators tune the structure and size of the problems. Typically, the parameters vary the number and type of world objects and the number of goals to achieve. It should be noted that the parameters of the generators are domain-specific, so they cause variations in complexity that differ across domains. Using these generators, reduced test sets of problems were created for solving by planners that were state of the art at recent IPCs: LAMA-2008 [29], LPG [30], and FF [31]. Problems with an increasing number of objects and goals were generated for solving within a 300-s time limit. As a general rule, the easiest problems of IPC-2011 had similar characteristics to those that were solved in tens of seconds in these limited runs. The most difficult problems were similar to those that were not solved in 300 s by LAMA-2008, LPG, and FF. Therefore, tuning of parameters to generate the competition problems was mainly a trial-and-error procedure. In fact, in some domains (such as FLOORTILE and VISITALL), the entire set of problems had to be regenerated several times as they were either too easy or too difficult for the IPC-2011 planners. This approach heuristically avoided a blind search in the parameter space for the problem generators.

Unavoidably, the selection of planners cited above introduces a bias towards those instances that are solvable by precisely those planners, which favors entrants that implement similar approaches, such as heuristic planning. However, there is no guarantee that they will be equally hard for planners that implement different techniques, such as SAT. Specifically, if planner Y solves a domain D much more effectively than planner X, then the performance delta between Y and X will appear small if most instances from D are selected such that X can still solve them; by contrast, if X is much better in D and most instances from D are selected such that X can still solve them, then the delta to Y will appear very large. Therefore, if other planner types had been used to test the problems, the particular planning tasks chosen would have been different. An illustrative example is the case of MADAGASCAR-P, which does not implement heuristic search and solves all instances in the PEGSOL domain in less than 4 s, far faster than its competitors. Once the competition was over, its author fixed a couple of bugs that improved its performance dramatically, so that all instances in the FLOORTILE domain could be solved in less than 1 s, whereas other planners such as LAMA-2011 and PROBE solved only six and five instances, respectively. Other domains for which the corrected version of MADAGASCAR-P excelled are PARCPRINTER and WOODWORKING. As additional evidence of the bias introduced by our selection mechanism, LAMA-2011 and PROBE solved either 19 or 20 planning tasks in

[6] http://www.plg.inf.uc3m.es/ipc2011-deterministic/NonUsedDomains.



(a) Sequential optimal track (b) Sequential satisficing track

(c) Sequential multi-core track (d) Temporal satisficing track

Fig. 1. Problems solved by different numbers of competitors in the four tracks of IPC-2011. The *y*-axis denotes the number of problems solved and the *x*-axis shows the number of planners. Three series are shown: the number of problems solved by exactly *x* planners; cumulative values; and the number of problems solved by at least *x* planners.

the domains BARMAN, PARKING, and VISITALL, whereas neither MADAGASCAR nor MADAGASCAR-P succeeded in solving a single instance. Summarizing, if harder instances would have been selected from the domains PEGSOL, FLOORTILE, PARCPRINTER or WOODWORKING the delta performance of MADAGASCAR-P would have been more noticeable. On the opposite side, the selection of problems from the domains BARMAN, PARKING and VISITALL clearly harmed the assessment of the performance of MADAGASCAR-P. Appendix C.1 provides further details.

For domains used in the previous competition, we used publicly available data to rank planning tasks in ascending order of expected difficulty. Again, the resulting ordering should be expected to be significantly biased towards the planners used in IPC-2008, although the bias should be expected to be smaller because the number of planners used is greater and there is more heterogeneity. There are many ways to rank the expected difficulty of planning tasks, the most evident one being the number of planners that solved them. We tried to use a more informative measure that took into account which specific planners solved which problems based on the Glicko rating system [32]. Appendix D gives more information about the procedure followed.

Fig. 1 shows the number of planners that solved a particular number of problems considering the whole benchmark, that is, those that were reused according to the procedure described in Appendix D and the new problems. As a result of the methodology used, we expected the results to fit a normal distribution. In other words, we expected many problems to be solved by several planners and fewer problems to be solvable by only a few planners or all of them, corresponding to the two tails of a normal distribution. These desirable properties were observed for all the IPC-2011 tracks except the sequential optimal track. In this case, many problems were not solved by any competing planner, and many problems were solved by many entrants. In our experience, selecting problems for this track seems to be very challenging, mainly because planners scale worse in this track, which reduces the maximum level of problem complexity they can handle. Thus, selection of structurally different problems becomes harder and problems tend in general to be more similar in terms of difficulty than in the other tracks: either they are not solved at all or a large number of planners solve them, as evidenced, for example, by the results for the BARMAN domain.

4. Results

Apart from the *quality* of the solutions, typical parameters used to compare planner performance in previous IPCs include the *number of instances solved* (or, alternatively, the *overall problem solving success rate*, defined as the percentage of problems solved[7]), and the raw *speed* at which solutions were generated. In this section we consider all these parameters and an additional one, *memory management*. Examining memory usage is important. Most planners go through a preprocessing step that instantiates all operators and predicates and incurs memory overheads, which can worsen significantly when taking into account that best-first search strategies are widely used. In addition, some planners can create particular structures to guide the search, such as BDDs [34], or abstractions that have high memory demands [35]. Finally, since a planner can return invalid solutions, VAL [4] was used and the percentage of invalid solutions is also reported for all tracks.

As well as figures showing results for coverage, quality, raw speed, memory usage, and the percentage of invalid plans, the analysis involved a number of statistical tests. A detailed description of these tests is in Appendix E. We tried to avoid the phenomenon known as *p-hacking* by explicitly reporting how we determined our sample size, all data exclusions, all manipulations, and all measures in the study [36]. Sample selection is explained in Section 3 and Appendix B; data exclusions and manipulations are discussed below; and measures are detailed in the subsection for each track.[8] The results for the statistical tests should be interpreted as a *measure of effort* with regard to the benchmarking and parameter selection, and thus they cannot necessarily be easily generalized to other problems or parameters. For this, further studies should be conducted such as reporting effect sizes, confidence intervals or looking at the variance of performance using bootstrap resampling. In other words, they are used solely to support the final conclusions for particular benchmark and parameter selections.

An important observation is that all statistical tests involve pairwise comparisons of two samples. This raises the question of how to deal with cases in which an entry has an unknown value. Although this is not the case when comparing the number of instances solved (since each planning instance is assigned a value per planner: it is either solved or not) or memory usage (even if a planner does not succeed in solving a problem, memory consumption can still be monitored), this situation arises when observing quality and raw speed, since one planner might not solve a particular planning instance whereas the other does. This issue has been managed in a variety of ways in the past and in the following we refer to the analysis conducted for the third and fifth IPCs.[9]

One solution consists of assigning an arbitrarily bad quality score or runtime to instances that were not solved. Another solution involves conducting alternative statistical tests but considering only *double hits*, or pairwise associations for which both values of the same variable (either quality or runtime) refer to solved instances. In the third IPC, double hits were used in addition to ordinary analysis when studying plan quality; when observing runtime, the organizers assigned an infinitely bad speed to the planner that did not solve a particular case [37]. The organizers of the fifth IPC adopted a different, but similar, methodology: when comparing runtime they assigned twice the time limit to cases for which no solution was generated, observing that this was the minimum value for which the performance gap for a problem solved by one planner and not solved by the other is greater than the performance gap for any problem solved by both. In their statistical analysis of plan quality, they partly adhered to the same recommendations made earlier and considered only double hits [38].

In our analysis of the IPC-2011 results, we tried to encompass both approaches, although there are some differences. When analyzing raw speed we conservatively assigned 1800 s (the maximum time allotted for solving each instance) to cases that were unsolved. This amounts to assuming that every planner eventually solves every instance. This diminishes the effect of coverage and reduces the chance of obtaining a significant difference, which is a desired effect since the study refers only to raw speed. However, when examining plan quality we considered only double hits. The reason is that in preliminary analysis conducted over all pairs of problems (and assigning infinitely bad quality to cases that were unsolved), we observed a significant influence of coverage, which dramatically favored participants that solved more problems. Indeed, we observed a nearly perfect correlation between coverage and the ordering suggested by the results of that analysis.

4.1. *Format of the results presentation*

The following subsections introduce the main analysis findings for the results in each track. Each subsection starts by analyzing coverage and the official IPC metric. When examining the number of instances solved, the evolution over time is plotted to provide empirical evidence of the influence of the time bound. These data provide an overall assessment of the ability of planners to produce fast responses if plan quality is ignored. The profile of these curves increases every time a new solution is found. Therefore, they provide an indication of whether a planner solves more problems at any particular time instant, although they give no information about the particular instances been solved. In addition, a steep slope indicates that a planner is able to solve plans at a fast pace in a short time.

Raw speed and memory usage are then examined in detail. Typical memory management strategies are discussed and exemplified for selected cases. The distribution of (peak) memory usage is also plotted as a function of the number of planning instances. These plots were generated considering only instances that were successfully solved. This curve is more informative because the last point on the *x*-axis represents the coverage of every planner and it provides a better view of the number of problems that each planner can be expected to solve for a given amount of memory. It should be noted,

[7] In the fourth IPC, research teams were allowed to decide what problems to attack. Thus, the summary of results also includes the *attacked ratio* [33]. IPC-2002 also prominently featured something akin to an *attacked ratio*. However, it is ignored here, since all entrants were faced with all problems in this edition of the IPC.

[8] A detailed description is also available in the RESULTS section of the competition website (http://www.plg.inf.uc3m.es/ipc2011-deterministic/Results).

[9] By contrast, the organizers of the fourth IPC made manual comparisons by examining the resulting data.

Table 4
Number of problems solved and success rate for the sequential optimal planners.

	FDSS-1	FDSS-2	SELMAX	M&S	LMCUT	FD-AUTOTUNE
Solved	185	182	169	169	167	166
Success rate	66.07%	65.00%	60.35%	60.35%	59.64%	59.28%
	FORKINIT	BJOLP	LMFORK	GAMER	IFORKINIT	CPT4
Solved	158	151	148	148	144	44
Success rate	56.42%	53.92%	52.85%	52.85%	51.42%	15.71%

however, that measuring memory usage is not straightforward and a number of subtleties should be taken into account. On one hand, Linux implements a lazy allocation/deallocation policy that might produce false results from time to time, and this observation applies to all tracks. For planners that start various processes (and the sequential multi-core in particular), the IPC-2011 software computed the memory used by one particular planner as the sum of the memory used by all of its subprocesses, even if they are sharing data, such as libraries or private data. In addition, the limit imposed on memory usage for one process is inherited by its subprocesses, along with all the descriptors used by the parent, but the operating system allows the newly generated process to allocate space on a different segment provided that no process exceeds its own limit. However, the overall memory usage can exceed the original limit. These observations are particularly important, as shown in Sections 4.3.3 and 4.4.3 when analyzing data generated in the sequential satisficing and multi-core tracks, respectively.

The results for different statistical analyses are presented in digraphs that show partial orders of dominance between planners. In these graphs, each vertex is a planner and there is an edge between planner A and B if and only if A statistically dominates the performance of B with regard to the random variable under study for a given confidence level. When dominance is detected at confidence level $\alpha = 0.001$, a solid edge is drawn. Alternatively, if dominance is found at a less restrictive confidence level of $\alpha = 0.005$, the edge is dashed. The absence of a simple path from A to B indicates that no statistically significant relationship was found between them and therefore that no transitive ordering can depend on such a relationship. When considering all instances, the resulting dominance graphs are necessarily acyclic since all edges refer to pairwise comparisons over the same benchmarking set. However, this is not true when using only double hits, and acyclicity cannot be enforced. All these statistical tests are described in Appendix E.

Finally, each subsection ends with a justification of the winner and runner-up selected in each track.

4.2. Performance of the sequential optimal planners

As mentioned in Section 2.3, no optimal specific solver was developed for any domain and thus there is no assurance that the solutions provided were always optimal. However, it was noted that the solutions for each instance always had the same total cost, and this was true for all planners that successfully solved the instances. Therefore, it is reasonable to assume that all planners behaved as expected and found optimal solutions.

4.2.1. Number of problems solved and quality

Table 4 lists the number of problems solved and the success rate for each planner in descending order. Since we checked that all correct solution plans had the same quality, the score assigned to each planner and planning task can only take a value of 0 (unsolved or invalid) or 1 (optimally solved). Hence, the final score for each planner equals the number of problems it solved. Therefore, Table 4 also shows the final score for every entry. In this track, invalid solutions were almost never generated apart from a couple of exceptions: CPT4 generated three invalid solutions in the PEGSOL domain and one in the SOKOBAN domain.

Fig. 2 shows results for the statistical test on coverage. The results differentiate four groups. The first group comprises the two top-ranked planners, which solved 185 and 182 problems, respectively. The second group consists of planners that were able to solve between 158 and 169 problems, and GAMER, which solved 148 instances. The third group contains planners that solved between 144 and 151 tasks, except for GAMER. The last group comprises only the CPT4 planner, which solved 44 problems. The results are not surprising and endorse the idea that optimal planners that solve more problems are superior; however, some insights are obtained. There are no significant differences (at the highest confidence level) in the performance of planners in the same group; this is attributed to chance when using a high confidence level. In fact, when comparing the number of problems simultaneously solved by two planners in the same group, it was found that either the difference is too small to be significant (e.g., FDSS-1 solves the same problems as FDSS-2 and only three additional ones) or is slightly larger but the intersection is rather low (e.g., GAMER and FORKINIT solve 148 and 158 problems, respectively, of which 119 are solved by both). This is not the case when comparing planners in different groups. The usual case is that one planner dominates the other and solves a high percentage of the problems solved by the other; for example, SELMAX solved 169 problems, of which 167 were solved by FDSS-2, while FDSS-1 solved them all.

4.2.2. Analysis of CPU time

Fig. 3 shows the evolution of the number of problems solved by all entrants in the time interval $(0, 1800]$ s. Although SELMAX performs worse than a number of planners at the start, it performs as well as MERGE-AND-SHRINK at the end of the interval and moderately better than LMCUT and FD-AUTOTUNE. However, all these planners, along with IFORKINIT, perform

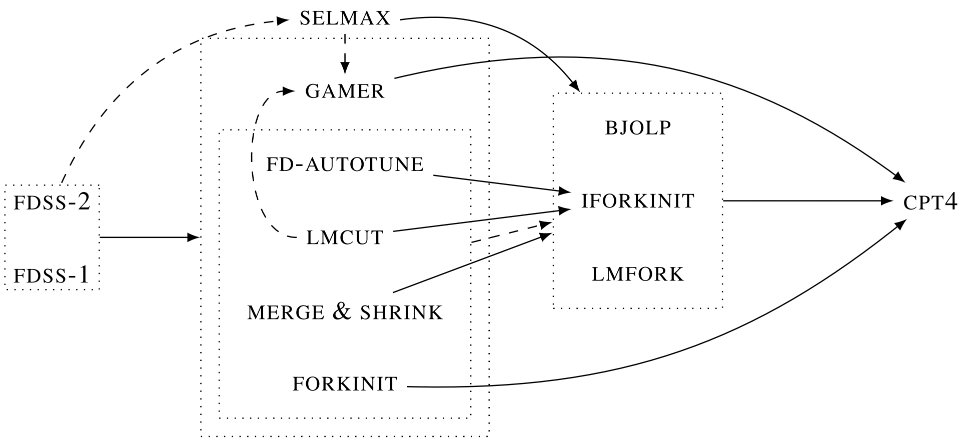
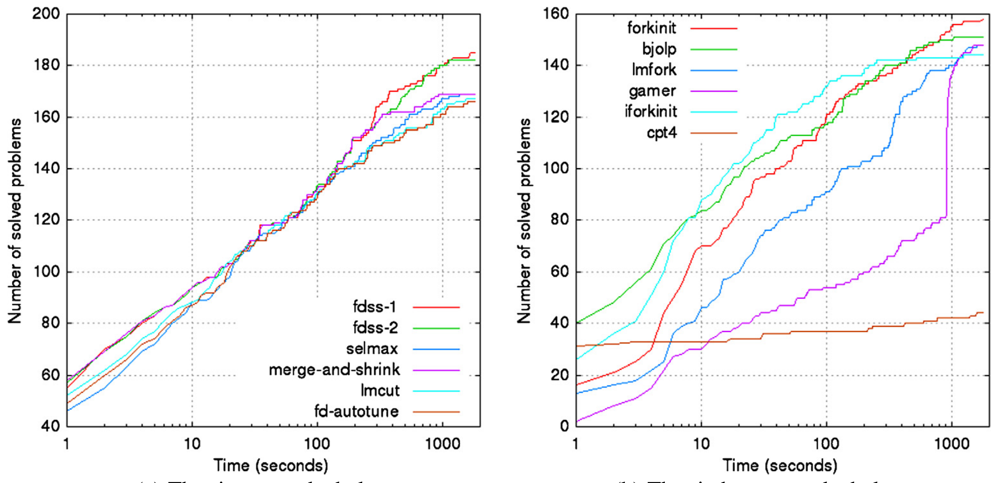


Fig. 2. Partial order for planner performance in the sequential optimal track in terms of successfully solved problems according to the binomial test with $p = 0.5$.



(a) The six top-ranked planners (b) The six bottom-ranked planners

Fig. 3. Evolution of the number of problems solved over time in the range $[1, 1800)$ s. The x-axis is shown on a logarithmic scale.

more or less equivalently between 20 and 200 s, with the latter showing a comparative decrease in performance after the first 100 s. In particular, it falls below the two variants of FDSS at $t = 122$ s and clearly below the group of leading planners in performance at $t = 251$ s. The case of GAMER is untypical. It shows a spectacular improvement in score in the middle of the time range considered. The reason is that this planner spends half of the available time in non-unit cost domains building a pattern database (PDB) (with abstraction in some cases and without in others) with a symbolic backwards breadth-first search. If the start state is found while building the PDB and no abstraction was used (i.e., a perfect mapping is used instead) the optimal solution is displayed immediately. Otherwise, more searching is needed and when the 15-min period is over, the planner starts a symbolic forward search from the initial state using the information in the PDB. Hence, it is very tempting to regard all problems solved by GAMER during the first 900 s as relatively easy. At $t = 896$, GAMER had solved 81 problems, and then it solves 59 additional tasks in just 163 s, performing equally well as LMFORK from this timepoint onwards.

Fig. 4 shows the results of the Wilcoxon signed-rank test for run time. The two variants of FAST-DOWNWARD STONE-SOUP are considered to be the fastest, followed by MERGE-AND-SHRINK. A plausible explanation is that this is an effect of the coverage of these planners since the statistical test considers all instances. An alternative statistical test conducted using only double hits concluded that FDSS-1 is faster than MERGE-AND-SHRINK, which is faster than FDSS-2. This is a counterintuitive result, since these planners are portfolios that consist of a collection of solvers that are invoked in succession. However, although SELMAX was able to solve problems at faster pace than other planners, especially in the second half of the allotted time, LMCUT and FD-AUTOTUNE (which solved roughly the same number of problems) were faster than it at a confidence level of $\alpha = 0.001$. The reason is that during the first 30 s, LMCUT and FD-AUTOTUNE solve problems faster than SELMAX does, and the number of problems solved in this interval amounts to 66.3% of the problems solved by the end. From this point on, SELMAX solves slightly more problems than FD-AUTOTUNE does, but it is not until $t = 80$ that SELMAX catches LMCUT, with

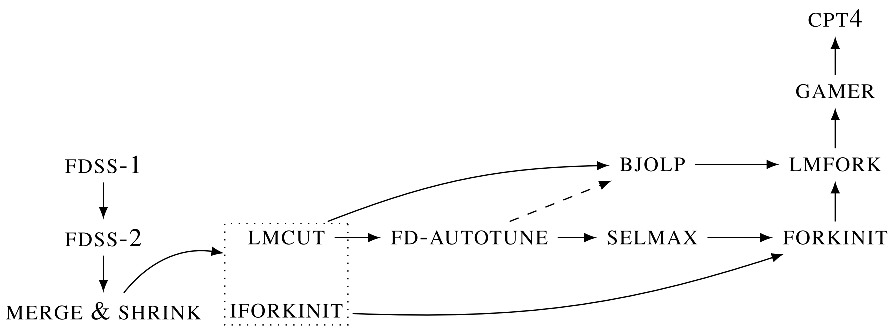


Fig. 4. Partial order for planner performance in the sequential optimal track for time taken to find the solution according to the Wilcoxon signed-rank test.

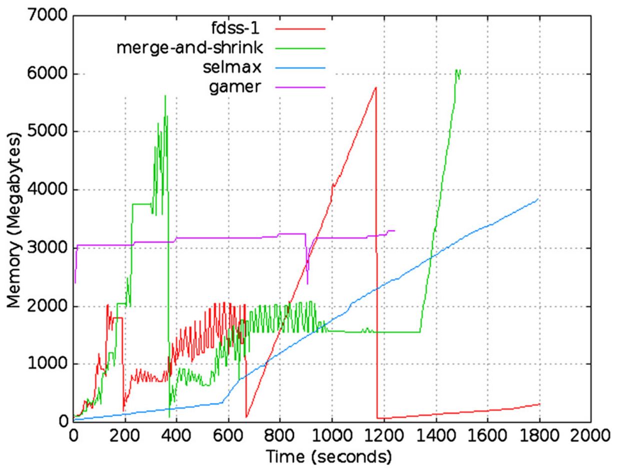


Fig. 5. Examples of typical memory profiles observed in the sequential optimal track. The figure shows the memory profile of the portfolios FDSS-1 and MERGE-AND-SHRINK, and the planners SELMAX and GAMER for solving problem 018 of the WOODWORKING domain.

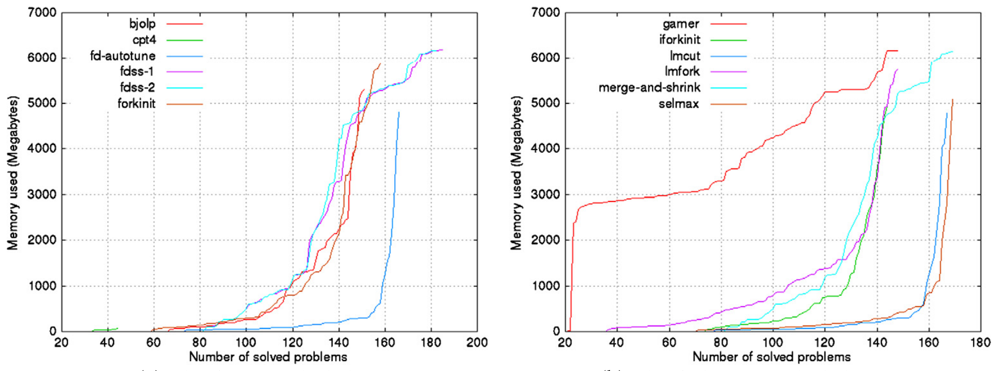
more than 75% of the problems solved. The case of GAMER, as anticipated above, is particularly interesting and the Wilcoxon signed-rank test indicates that it is slower than all the other entrants except for CPT4.

4.2.3. *Analysis of memory performance*

Typically, planners use memory incrementally according to a monotonically increasing profile. However, other strategies for memory management have been identified. With the advent of portfolios (such as the FAST-DOWNWARD-STONE-SOUP family and MERGE-AND-SHRINK), memory can be used and released according to the replacement policy implemented. In addition, some planners implement particular strategies to create heuristics that are used later to guide the search. This is the case for GAMER and for MERGE-AND-SHRINK, which augments the size of a table with admissible estimates until it reaches a particular limit. At this point, the table is shrunk and the search proceeds. This strategy results in a memory profile that shows some decreases at the points at which shrinking is performed.

For illustration purposes, Fig. 5 shows the memory profile for four different planners in solving problem 018 of the WOODWORKING domain. The planners considered are FDSS-1, MERGE-AND-SHRINK, GAMER, and SELMAX. Their selection is justified because they all implement the three different strategies identified above. From the preceding figure, another diagnosis can easily be inferred: MERGE-AND-SHRINK performs a number of shrinking operations, the most prominent one at $t = 370.59$ s, at which the memory decreased from 5277.31 MB to 91.70 MB in just 5 s. FDSS-1 shows the typical memory profile of a portfolio whereby memory is used and released as the solvers implemented in it are successively invoked. It is notable that the second solver used by FDSS-1 in this particular case shows a memory profile much like that of MERGE-AND-SHRINK. Inspection of the code for FDSS-1 revealed that this is exactly the second planner that is automatically invoked by it. SELMAX uses memory incrementally, as most planners do, and always keeps within the limits. Indeed, it has been observed that SELMAX incurs a very low percentage of memory failures. Finally, GAMER, the only planner that succeeded in solving this particular planning task, shows a monotonically increasing profile while building the PDB up to $t = 900$ s. At this point, the memory usage falls abruptly. This is because the PDB creation process is killed and the memory allocated is freed. From here, another search is conducted forward until the solution is found at $t = 1237.86$ s with memory consumption of 3303.72 MB, slightly greater than half of the available memory.

Fig. 6 shows the memory required to solve a particular number of tasks.



(a) The six top-ranked planners (b) The six bottom-ranked planners

Fig. 6. Maximum memory (y-axis, MB) taken by each entry in the sequential optimal track for solving a specific number of planning tasks.

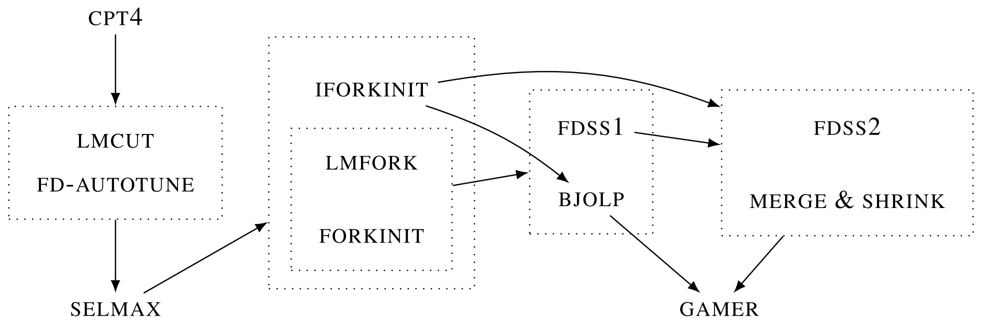


Fig. 7. Partial order of the performance of planners for memory usage in the sequential optimal track according to the Wilcoxon signed-rank test.

The Wilcoxon signed-rank results for memory usage are shown in Fig. 7. Surprisingly, in spite of all the available memory (up to 6 GB of RAM), there are four planners (CPT4, LMCUT, FD-AUTOTUNE and SELMAX) that take, on average, less than 1 GB. Planners that are known to use massive memory are ranked last according to this metric: GAMER, which creates BDDs, and MERGE-AND-SHRINK, which creates abstractions. Finally, FDSS-1 uses much more memory, on average, than the other planners, and this observation is supported by the statistical test results, yet it never fails on memory. The reason is that it monitors its own memory usage. When it exceeds a given limit, the solver is aborted and the next one is invoked.

4.2.4. Distinguished performers in the sequential optimal track

The results in Table 4 (page 92) reveal the following:

- FDSS-1 and FDSS-2 were the top performers among all entries in the sequential optimal track, as evidenced by the statistical test results summarized in Fig. 2, page 93.
- Among all entries shown in the second level of Fig. 2, SELMAX and MERGE-AND-SHRINK solved more problems.

Therefore, the following planners were distinguished by their performance in the sequential optimal track of IPC-2011:

- *Winner*: instead of nominating two variants of the same planner as winners, the one that solved more problems was chosen as the winner. The planner selected was FDSS-1, which solved 185 problems in total.
- *Runner-up*: although there is no statistical evidence (at a confidence level of 99.5%) of differences between SELMAX, MERGE-AND-SHRINK, FD-AUTOTUNE, LM-CUT, and FORKINIT, the first two were chosen as joint runners-up because they solved more problems, 169 each in total.

An interesting observation is that two of the three winning planners (FDSS-1 and MERGE-AND-SHRINK) are portfolios. Moreover, the three portfolios in the competition (including FDSS-2) showed the best performance among all entries, tying only with SELMAX.

4.3. Performance of the sequential satisficing planners

For the sequential satisficing track there were 27 competing planners. The following subsections examine their performance under different parameters.

Table 5
Score, number of problems solved, and success rate for the sequential satisficing planners.

	LAMA-2011	FDSS-1	FDSS-2	FD-AUTOTUNE-1
Score	216.33	202.08	196.00	185.09
Solved	250	232	233	223
Success rate	89.28%	82.85%	83.21%	79.64%
	ROAMER	FD-AUTOTUNE-2	FORKUNIFORM	PROBE
Score	181.47	178.15	177.91	177.14
Solved	213	193	207	233
Success rate	76.07%	68.92%	73.92%	83.21%
	ARVAND	LAMA-2008	LAMAR	RANDWARD
Score	165.07	163.33	159.20	141.43
Solved	190	188	195	184
Success rate	67.85%	67.14%	69.64%	65.71%
	BRT	CBP2	DAE_YAHSP	YAHSP2
Score	116.01	98.34	95.23	94.97
Solved	157	135	110	138
Success rate	56.07%	42.85%	39.28%	49.28%
	YAHSP2-MT	CBP	LPRPGP	MADAGASCAR-P
Score	90.95	85.43	67.07	65.93
Solved	132	123	118	88
Success rate	47.14%	43.92%	42.14%	31.42%
	POPF2	MADAGASCAR	CPT4	SATPLANLM-C
Score	59.88	51.98	47.85	29.96
Solved	81	67	52	32
Success rate	28.92%	23.92%	18.57%	11.42%
	SHARAABI	ACOPLAN	ACOPLAN2	
Score	20.52	19.33	19.09	
Solved	33	20	20	
Success rate	11.78%	7.14%	7.14%	

4.3.1. *Number of problems solved and quality*

According to the expression given in Section 2.3, each planner is assigned a score in the interval $[0, 1]$, with 0 if no solution is found and 1 if no planner found any solution with better quality or, alternatively, with a lower total cost. Therefore, coverage (or the number of problems solved) does not necessarily equal the final score. Table 5 shows the final score, the number of problems solved, and the success rate in descending order for the first parameter. This table considers only solutions validated by VAL. For this track, 13.6% of the solutions were deemed invalid by VAL. It should be noted that this happened a number of times just because the output was not compliant with the format recognized by VAL.[10]

It is evident that LAMA-2011 ranked first in terms of both the number of problems solved and the scoring function used. FDSS-1 solved just one fewer problem than PROBE and the FDSS-2 variant. Although the correlation between coverage and score in this track is as high as 0.98, a few planners would have been ranked differently if coverage had been chosen instead. Remarkably, PROBE ranks third for the number of problems solved (tied with FDSS-2) instead of eighth, whereas FD-AUTOTUNE-2 ranks ninth for the number of problems solved instead of sixth. These differences are due solely to the quality of the solutions found. The explanation is that although PROBE has excellent coverage, its plans tend to be worse than those generated by other entrants. Similarly, FD-AUTOTUNE-1 does not solve as many problems as its final position might suggest, but its plans are usually better.

Fig. 8 shows the partial order according to the binomial test results for coverage. Likewise, Fig. 9 (page 98) shows the partial order according to the Wilcoxon signed-rank test for plan quality. Recall from the beginning of this section that only double hits are considered in the second case. It should be noted that some results suggest a significant difference in favor of a planner that solves far fewer problems. This is reasonable since use of only *double hits* completely ignores the overall performance of each planner. A plausible interpretation of this observation is that some planners might solve problems with higher quality solutions but that the price they pay in searching to find these solutions is so high that they solve a much smaller set of problems.

In terms of quality, CPT4, DAE_YAHSP, FDSS-1, and ARVAND dominate all the other planners at a confidence level of 99.9%, and CPT4 and FDSS-1 are the only non-dominated planners at a confidence level of 99.5%. Importantly, PROBE is dominated by a good number of planners, including LAMA-2011, both variants of FAST-DOWNWARD STONE-SOUP, and FD-AUTOTUNE. This endorses the intuition that the quality of plans generated by PROBE is low. Interestingly, when coverage is fully ignored, LAMA-2011 appears to be dominated by a number of planners: CPT4, FD-AUTOTUNE-2, and FDSS-1 at a confidence level of 99.9%, and additionally by ARVAND at a confidence level of 99.5%. Undoubtedly, the difference in score between LAMA-2011

[10] For example, according to its authors, ACOPLAN suffered from an internal bug that caused it to write some plans in an invalid format (Fabio Rossi, personal communication).

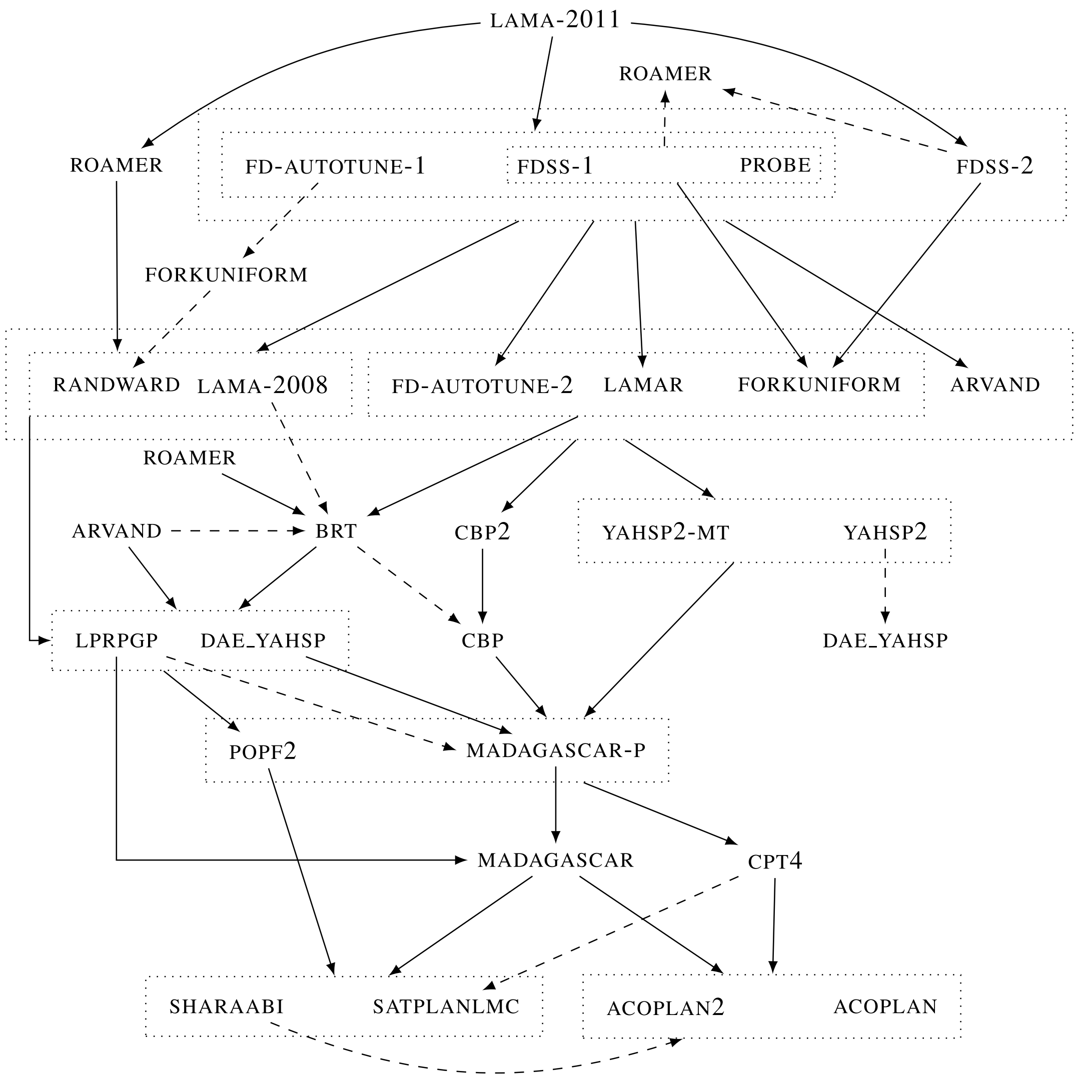


Fig. 8. Partial order for planner performance in the sequential satisficing track in terms of coverage according to the binomial test with $p = 0.5$. The nodes ARVAND and ROAMER have been replicated for clarity.

and these planners should mainly be attributed to coverage, since for every successful planning instance the score necessarily increases. With regard to the winner of the previous edition, LAMA-2008 still shows competitive performance as it is dominated by only a few planners.

4.3.2. Analysis of CPU time

In this section, we first show how the number of problems solved evolved over time in the range $(0, 1800]$ s. Owing to the large number of participants in this track, Fig. 10 (page 99) has been split into four parts.

LAMA-2011 is the fastest planner in solving problems. It is only dominated by PROBE in the first 2 s, and is ranked first over all other entrants immediately thereafter. Interestingly, PROBE, FDSS-1, FDSS-2, and FD-AUTOTUNE-1 show a profile that is still increasing close to the end of the time span, that is, the planners shown in Fig. 10(a) suffer no stagnation in terms of their performance. This is not the case for YAHSP2-MT, as shown in Fig. 10(c), which stagnates severely after the first 199 s. A good example of a planner that solves plans at a very fast pace in some intervals is BRT (as shown in Fig. 10(c)), whose runtime distribution is typical of a system that performs a lot of preprocessing before starting to search for a solution plan.

To confirm all these observations, a statistical test was performed for the time taken until a first solution is found. Fig. 11 (page 100) shows the resulting dominance graph. As anticipated by our preliminary analysis, LAMA-2011 significantly outperforms all other entrants in the sequential satisficing track in terms of running time. In addition, as suggested by the preliminary analysis derived from Fig. 10(a), PROBE, FDSS-1, and FD-AUTOTUNE-1 are very fast, and are dominated only by the

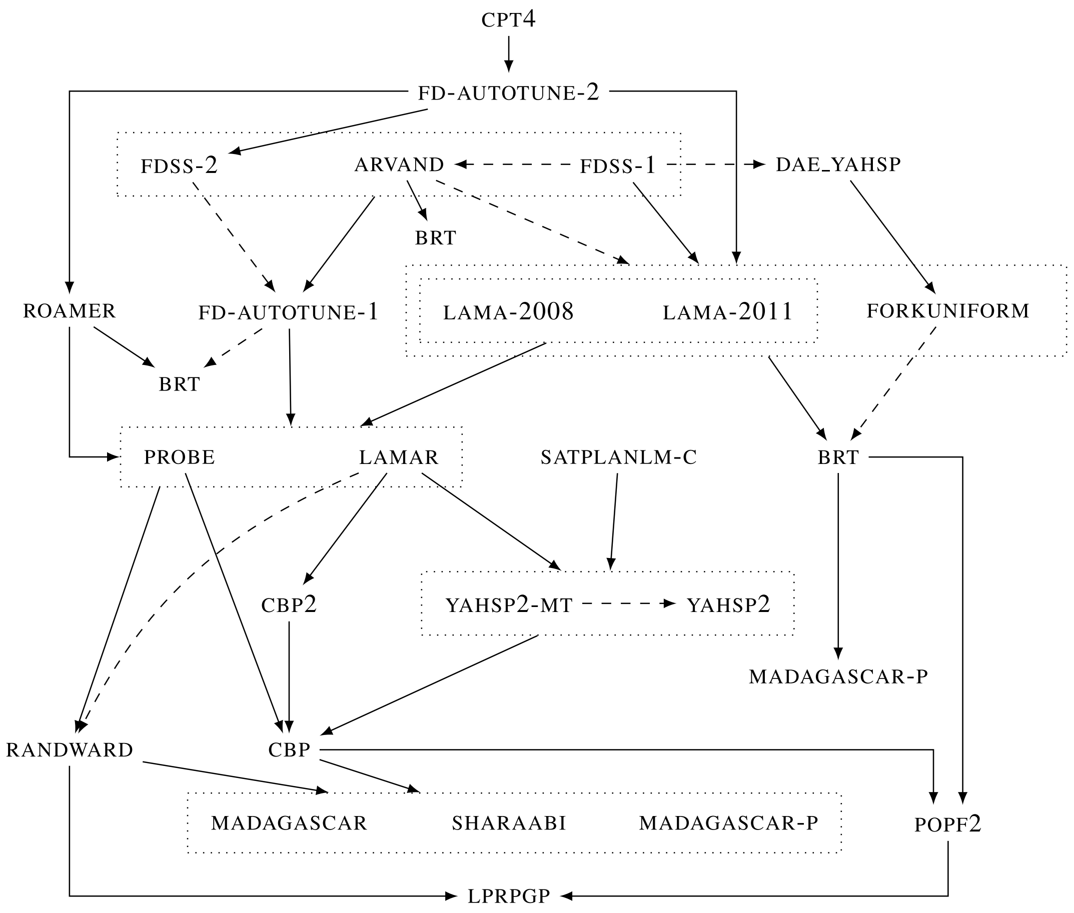


Fig. 9. Partial order for planner performance in the sequential satisficing track in terms of plan quality according to the Wilcoxon signed-rank test using only double hits. The nodes BRT and MADAGASCAR-P have been replicated for clarity.

winner, whereas FDSS-2 is dominated by the previous entries, the last two at $\alpha = 0.001$ and the first at $\alpha = 0.005$. Finally, it is worth noting that LAMA-2008 dominates half the competitors with regard to raw speed.

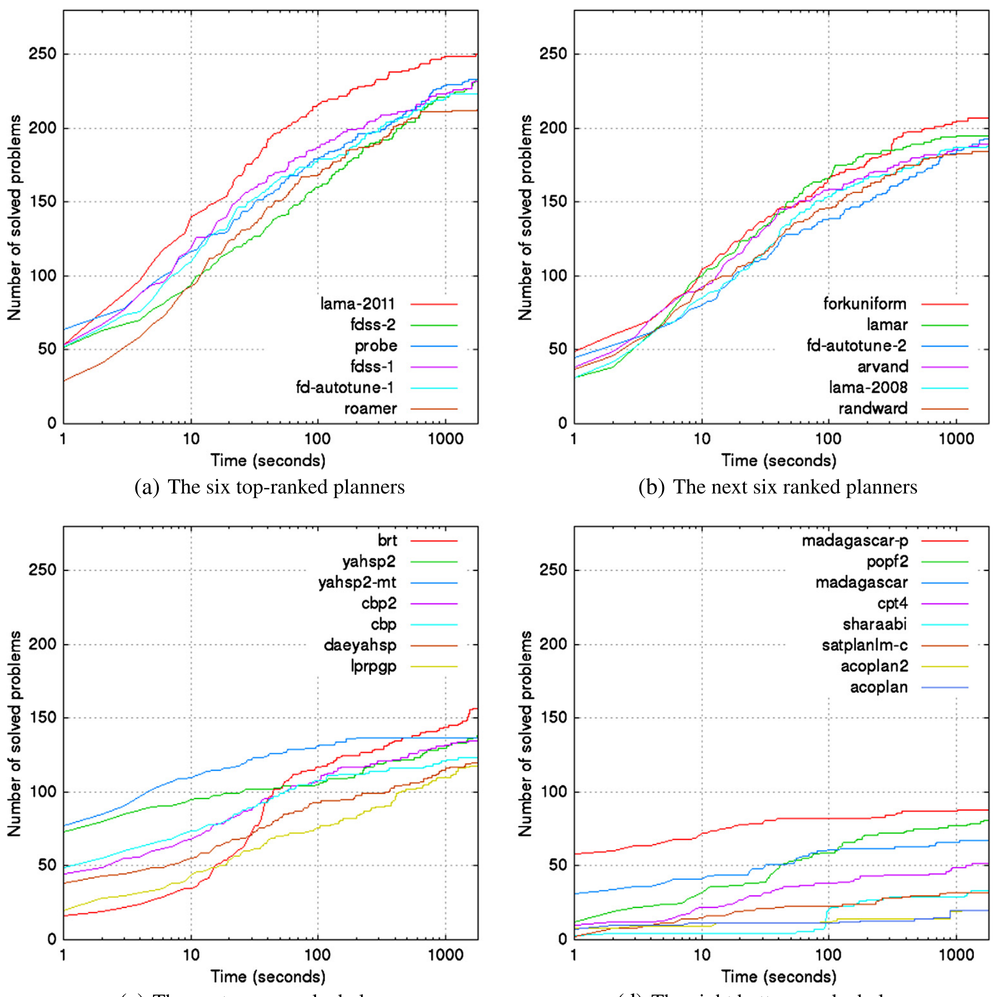
4.3.3. Analysis of memory performance

Since the evaluation schema of the competition emphasized quality by fixing the allotted time to 30 min, many planners followed one of two strategies: (i) most of them implemented anytime approaches that try to find a solution quickly and then refine it during the remaining time; and (ii) a few implemented portfolios or collections of different solvers that were invoked successively. In the first case, memory is usually consumed incrementally. In the second case, planners exhibit a memory usage profile that increases while one solver is trying to solve a particular problem, and then abruptly decreases (by releasing all the memory used) if a solution is not found before the next solver is started. This behavior is then repeated as many times as solvers are used.

For illustration purposes, Fig. 12 (page 100) shows an example of these strategies. It shows the memory profile for LAMA-2011 and the portfolios FDSS-1 and FDSS-2 in the 11th problem of the OPENSTACKS domain. All the planners successfully solved this case, with LAMA-2011 generating up to 18 different solutions and the portfolios generating seven solutions each. Whereas LAMA-2011 uses memory incrementally, FDSS-1 and FDSS-2 exhibit a number of cycles in which memory is used and then released.

Fig. 13 (page 101) shows the maximum memory required by all entrants in this track as a function of the number of problems solved by each of them. In particular, BRT takes less than 6 GB to solve 127 planning tasks, but it exceeds this limit to solve 30 additional tasks. BRT uses only a few hundreds of MB most of the time, but it can take a few GB when loading the results of a SAT compilation into memory. It has been observed that this task consistently lasts for less than 5 s and is performed by a subprocess that never uses 6 GB on its own.

Fig. 14 (page 102) shows the Wilcoxon signed-rank results for the maximum memory ever used. Not surprisingly, the worst-performing planners in terms of memory usage are among those that failed more often on memory: POPF2, LAMAR, LAMA-2008, RANDWARD, and ROAMER. By contrast, a few planners showed impressive management of available memory:



(a) The six top-ranked planners (b) The next six ranked planners

(c) The next seven ranked planners (d) The eight bottom-ranked planners

Fig. 10. Analysis of the evolution of the number of problems solved in the sequential satisficing track over time in the range $[1, 1800)$ s. The x-axis is shown on a logarithmic scale.

PROBE never failed on memory and DAE_YAHSP failed only once, although they are among the planners that use more memory. Finally, the portfolios FDSS-1 and FDSS-2 also managed to keep within the available memory and show good usage.

4.3.4. Distinguished performers in the sequential satisficing track

The results in Table 5 (page 96) reveal the following:

- LAMA-2011 ranked first according to the official score. It also showed outstanding performance in terms of both plan quality and raw speed, as evidenced by the statistical tests summarized in Figs. 9 (page 98) and 11 (page 100).
- FDSS-1 ranked second according to the official IPC score. When considering plan quality, FDSS-2 has a similar performance and CPT4 and FD-AUTOTUNE-2 are better, as suggested by Fig. 9. However, the second variant of FAST-DOWNWARD STONE-SOUP is dominated by the first one when observing raw speed, as shown in Fig. 11 (page 100), and CPT4 and FD-AUTOTUNE-2 rank far behind FDSS-1, as shown in Table 5.

Thus, the following planners were distinguished by their performance in the sequential track of IPC-2011:

- *Winner*: LAMA-2011, with an overall score of 216.33 and 250 problems solved.
- *Runner-up*: FDSS-1, with a final score of 202.08 and 232 problems solved.

Finally, it is worth noting that the analysis performed in Section 6 under alternative metrics shows that PROBE also exhibited outstanding performance. However, since the IPC emphasized plan quality, PROBE was not selected as a winner as quality was its major drawback.

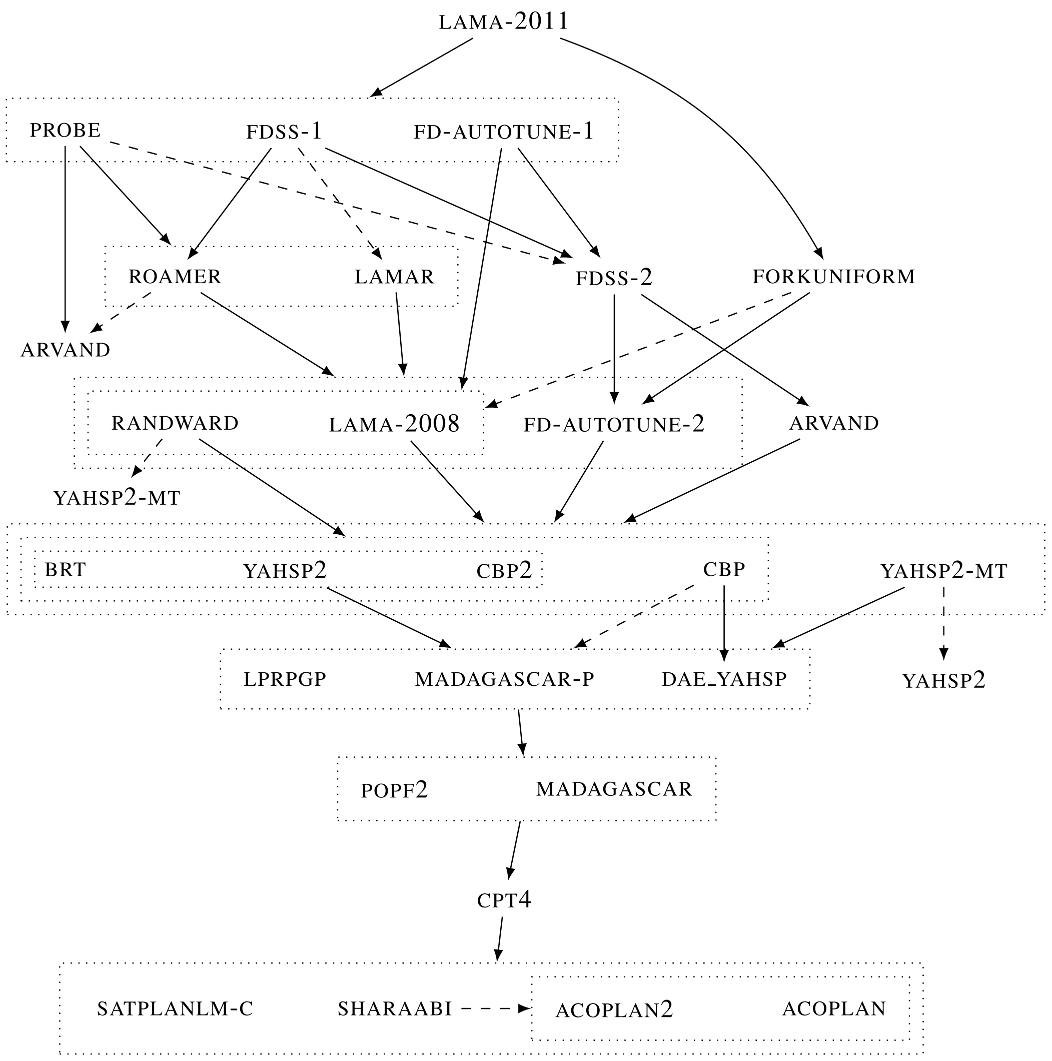


Fig. 11. Partial order for planner performance in the sequential satisficing track for the time taken to find the first solution according to the Wilcoxon signed-rank test.

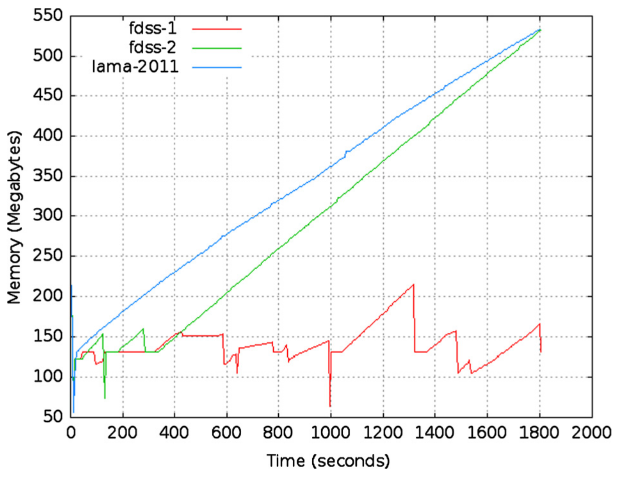


Fig. 12. Examples of typical memory profiles observed in the sequential satisficing track. The figure shows the memory profile for LAMA-2011 and the portfolios FDSS-1 and FDSS-2 in solving problem 010 of the OPENSTACKS domain.

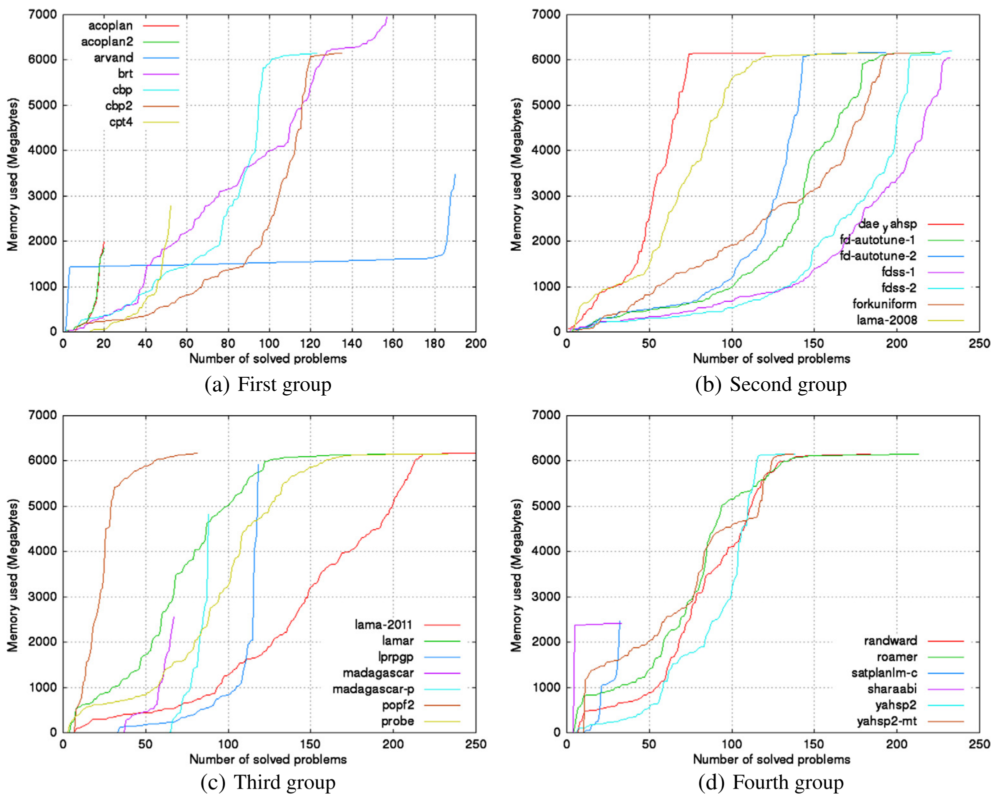


Fig. 13. Maximum memory (y-axis, MB) used by each entry in the sequential satisficing track in solving a specific number of planning tasks.

Table 6
Score, number of problems solved, and success rate for the sequential multi-core planners.

	ARVANDHERD	AYALSOPLAN	PHSFF	ROAMER-P
Score	227.07	159.95	130.59	129.06
Solved	236	184	163	140 (*179*)
Success rate	84.28%	65.71%	58.21%	50.0%
	YAHSP2-MT	MADAGASCAR-P	MADAGASCAR	ACOPLAN
Score	118.58	66.44	52.00	17.62
Solved	153 (*155*)	88	67	18
Success rate	54.64%	31.42%	23.92%	6.42%

4.4. Performance of the sequential multi-core planners

A sequential multi-core track was introduced in the IPC series for the first time in 2011 to acknowledge this planning architecture trend. Section 2.1 provides a description of its rules.

To facilitate comparisons among tracks, entrants in the sequential multi-core track were faced with exactly the same problems chosen for the sequential satisficing track; the results of this comparison can be found in Section 5.2.

4.4.1. Number of problems solved and quality

In this track, 16.5% of the plans issued were not valid according to VAL. Table 6 shows the final score for each planner, along with the number of problems solved and the success rate.

Remarkably, the winner of this track, ARVANDHERD, did not outperform the winner of the sequential satisficing track and it solved 14 fewer problems than LAMA-2011 did, despite being allowed more computational resources. A number of planners that entered this track also participated in the sequential satisficing track: YAHSP2-MT, MADAGASCAR-P, MADAGASCAR, and ACOPLAN. YAHSP2-MT was able to solve more problems when allowed to run the subprocesses for longer, solving 18 additional problems. MADAGASCAR and MADAGASCAR-P solved exactly the same number of problems, 88 and 67, respectively, and ACOPLAN, surprisingly, solved two fewer problems. The performance of MADAGASCAR and MADAGASCAR-P was as expected, since these are the same planners as entered in the sequential satisficing track. However, the case of ACOPLAN seems more difficult to account for. A plausible explanation is that this planner uses a stochastic algorithm so that small differences in coverage can easily be attributed to chance, even if more computational resources are allowed.

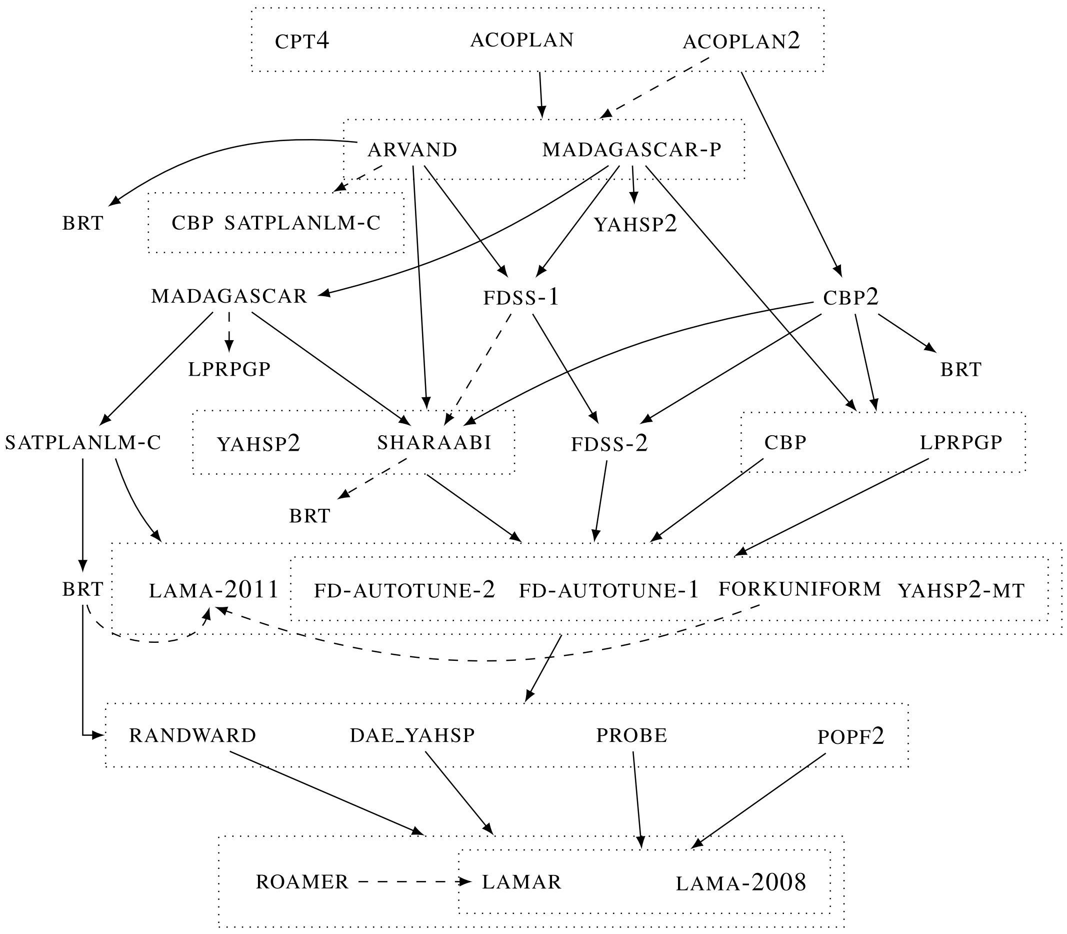


Fig. 14. Partial order for planner performance in the sequential satisficing track for the maximum memory ever used in each planning task according to the Wilcoxon signed-rank test.

As noted in Section 2.1, the IPC was run in a very strict mode. Among other considerations, if a planner ever produced an invalid solution, the problem was considered unsolved even if some valid plans were first generated. While this did not affect most planners across all tracks, two planners in the sequential multi-core track were affected by this decision. The first, YAHSP2-MT, created invalid solutions (along with valid solutions) in two different problems of the PEGSOL domain and thus these problems did not contribute to improving its score. By contrast, this decision severely hampered ROAMER-P, which created invalid plans in 39 different problems in the following domains (number of problems affected shown in parentheses): NOMYSTERY (1), BARMAN (7), PARKING (6), SCANALYZER (1), SOKOBAN (7), TIDYBOT (3), TRANSPORT (1), VISITALL (4), and WOODWORKING (9). ROAMER-P was equipped with a technique to avoiding plateaus based on random walks. After the competition, its authors discovered a bug in constructing new states that caused this undesired behavior, which decreased the number of problems effectively used for computing its score from 179 to 140. If no invalid problem had been issued in any of these cases, its final score would have been 165.01, ranking second in the end. For the sake of completeness, these values are shown in Table 6 in italics. In the following, we discuss the results for which generation of an invalid plan resulted in an invalid problem (even if there were some valid solutions also) unless stated otherwise.

From the preceding table it also follows that sorting the performance of the multi-core planners according to their score successfully ranks them with respect to coverage. There is only one exception: ROAMER-P solved 140 problems, whereas YAHSP2-MT solved 155, yet the former obtained a better score; in fact, the coefficient for correlation between coverage and the score was 0.986. A plausible explanation for this phenomenon lies in the quality of the plans found by each planner. The results in Table 6 suggest that the quality of the plans found by ROAMER-P are significantly better than those found by YAHSP2-MT. To test this conjecture and to provide additional information about the significance of the differences, various statistical tests were performed.

The first test analyzes the number of problems effectively solved by each planner. Fig. 15 shows the acyclic graph resulting from the binomial test for $p = 0.5$. If the cases for which ROAMER-P and YAHSP2-MT produced invalid solutions along with valid solutions are used favorably, the scenario changes significantly. The main differences are that the performance

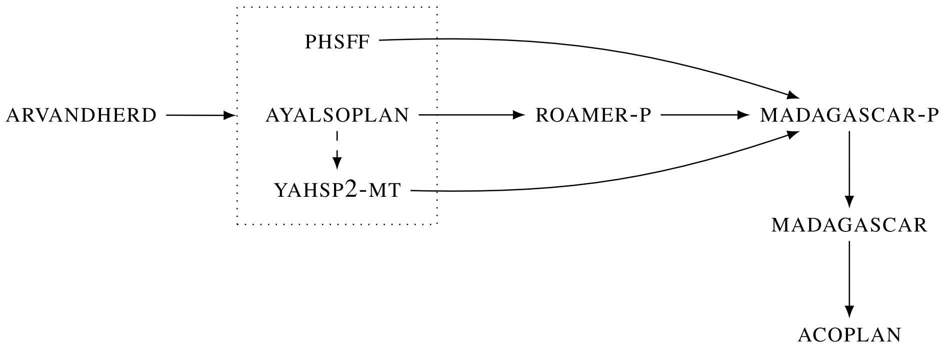


Fig. 15. Partial order for planner performance in the sequential multi-core track in terms of successfully solved problems according to the binomial test with $p = 0.5$.

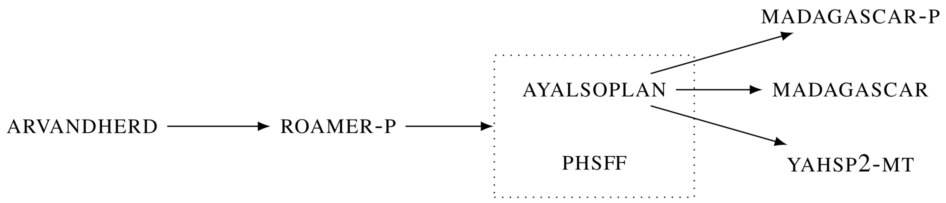


Fig. 16. Partial order for planner performance in the sequential multi-core track in terms of quality according to the Wilcoxon signed-rank test considering only double hits.

of AYALSOPLAN, PHSFF, ROAMER-P, and YAHSP2-MT is indistinguishable at a confidence level of 99.9%. At a confidence level of 99.5%, AYALSOPLAN dominates YAHSP2-MT, but not ROAMER-P or PHSFF.

Fig. 16 shows the dominance relationships between entrants in the sequential multi-core track with regard to plan quality as computed by the Wilcoxon signed-rank test. In this figure, ACOPLAN was removed because the set of problems solved was not considered to be statistically significant. As shown, ARVANDHERD finds plans with the best quality among all participants. The ability of ROAMER-P to find good plans as conjectured above is confirmed. Finally, there is no clear dominance among the other entries. Since the number of double hits between them is often just barely below 65% over the total number of problems solved (thus producing sample populations of reasonable size), this is attributed to the fact that they excel in disjoint sets of domains. Indeed, MADAGASCAR and MADAGASCAR-P are better in PARCPRINTER, whereas PHSFF clearly outperforms the other planners in ELEVATORS, SCANALYZER, SOKOBAN, and TIDYBOT; finally, YAHSP2-MT excels in BARMAN, FLOORTILE, TRANSPORT, and VISITALL. In other domains, either the performance is very similar (as in NOMYSTERY), or none of the planners can find solutions (especially in OPENSTACKS), so that no statistically relevant observations can be made.

4.4.2. Analysis of CPU time

Fig. 17 shows the evolution of the number of problems solved in the interval $(0, 1800]$ s. PHSFF and YAHSP2-MT are the fastest algorithms in the short term, with PHSFF always solving more problems than YAHSP2-MT up to the end of the interval. PHSFF solved 120 problems in the first 13 s (and YAHSP2-MT solved 109), whereas ARVANDHERD solved 119 problems. From this point on, the winner of this track progressed much faster and had already solved 123 problems at $t = 14$ s, and YAHSP2-MT solved only one additional problem. However, these planners are still among the fastest up to $t = 58$ s, at which point AYALSOPLAN catches YAHSP2-MT, with both planners solving 123 problems by this time. A couple of minutes later, at $t = 219$ s, the coverage of AYALSOPLAN equals that of PHSFF, with 153 problems solved, and it finally solves more problems, as shown in Table 6 (page 101).

Fig. 18 shows Wilcoxon signed-rank results for raw speed. The ability of PHSFF to find solutions promptly is much the same as for AYALSOPLAN and YAHSP2-MT at the most restrictive confidence level. At a confidence level of 99.5% the situation changes and PHSFF is faster than YAHSP2-MT, but not AYALSOPLAN. Other relationships shown in Fig. 18 are supported by the curves shown in Fig. 17. In particular, MADAGASCAR-P is faster than MADAGASCAR, which in turn is faster than ACOPLAN. There is, however, an interesting observation: Fig. 17 shows that ROAMER-P is clearly dominated by the two variants of MADAGASCAR up to $t = 1254$ s. By this point, ROAMER-P had solved 67 problems, the same number of problems solved by MADAGASCAR. A few minutes later, at $t = 1501$ s, ROAMER-P had solved as many problems as MADAGASCAR-P (88), and dominated MADAGASCAR and MADAGASCAR-P from this point on both. Indeed, at this time point it started to solve new problems very rapidly, reaching 140 problems, presumably as a result of the technique used to avoid plateaus (Section 4.4.1) since the planner is started once a preliminary search fails. This steep increment in performance is reflected in the Wilcoxon signed-rank test, which determines that ROAMER-P dominates both MADAGASCAR and MADAGASCAR-P.

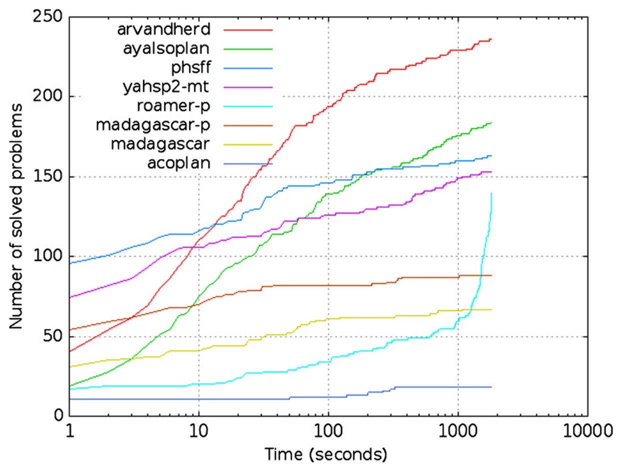


Fig. 17. Analysis of the number of problems solved over time in the range [1, 1800] s. The *x*-axis is shown on a logarithmic scale.

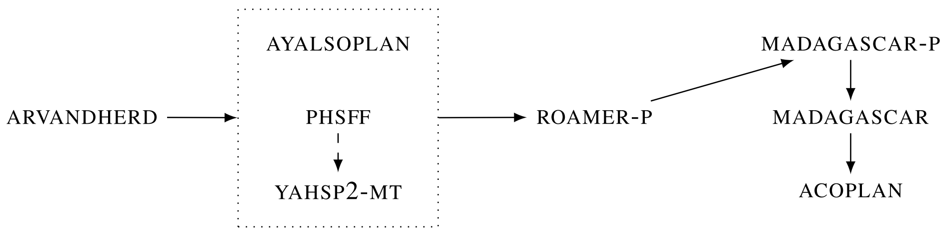


Fig. 18. Partial order for planner performance in the sequential multi-core track in terms of time taken to find the first solution according to the Wilcoxon signed-rank test.

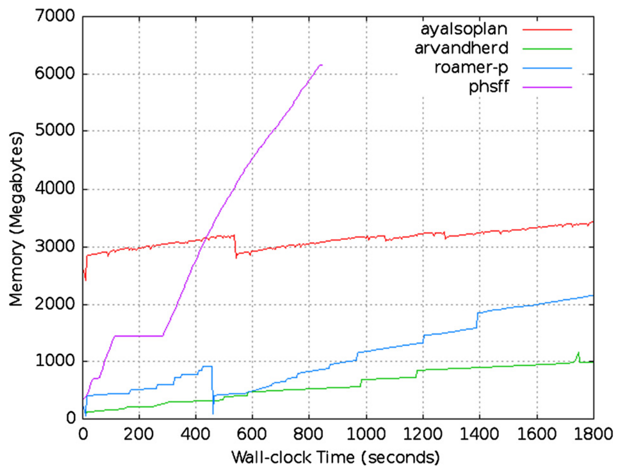


Fig. 19. Examples of typical memory profiles observed in the sequential multi-core track. The figure shows the memory profile for ARVANDHERD, AYALSOPLAN, PHSFF, and ROAMER-P in solving problem 012 of the TRANSPORT domain. The *x*-axis is shown on a logarithmic scale.

4.4.3. Analysis of memory performance

In the sequential multi-core track, all entrants launched a number of processes, each one consuming its own memory requirement. Since these processes can be started and terminated according to different policies, it is rather difficult to outline a general behavior of the sequential multi-core planners with regard to memory usage. For example, while most planners start a reduced number of processes and then launch threads when needed (e.g., PHSFF varies the number of threads from two and six for the same process), others (such as AYALSOPLAN) vary the number of both processes and threads during the whole processing time.

For illustration purposes, Fig. 19 shows the memory profile for ARVANDHERD, AYALSOPLAN, PHSFF, and ROAMER-P in solving problem 012 of the TRANSPORT domain. This figure differs slightly from Figs. 5 (page 94), 12 (page 100), and 27 (page 109).

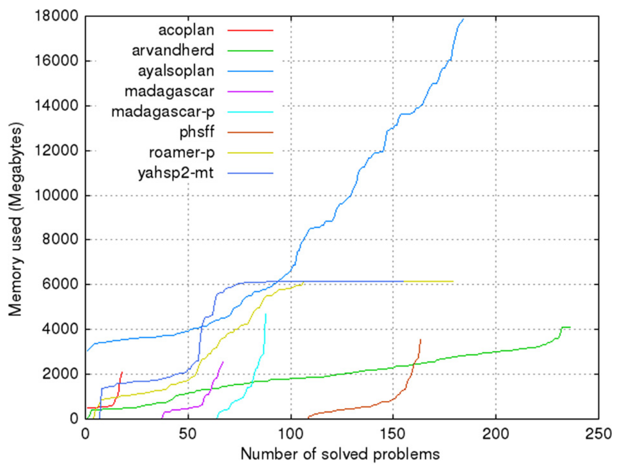


Fig. 20. Maximum memory (y-axis, MB) taken by each entry of the sequential multi-core track in solving a specific number of planning tasks.

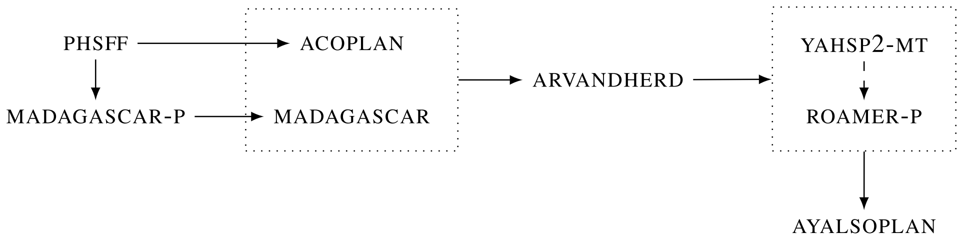


Fig. 21. Partial order for planner performance in terms of memory usage in the sequential multi-core track according to the Wilcoxon signed-rank test.

In this case, the x-axis shows wall-clock time instead of CPU time. In general, it was observed that planners in this track showed a monotonically increasing profile even if memory decreases from time to time.

Fig. 20 shows the maximum memory needed to solve a specific number of problems for all entrants in this track. The maximum memory reported by AYALSOPLAN results from various observations already made in Section 4.1. This planner starts a good number of processes, each of which stores a different hash table that is used as a pruning device. Although the memory required to store this table is tiny, every process consumes memory in storing the states that are encountered. Thus, every process takes additional memory without ever exceeding the maximum allotted, but in different segments, so that the overall memory goes well beyond the limit of 6 GB. To conclude, it is worth noting that YAHSP2-MT solves 81 problems and consumes more than 6000 MB in each execution, but always below 6 GB.

The Wilcoxon signed-rank results for the maximum memory ever used are shown in Fig. 21. Remarkably, ARVANDHERD almost never exceeded the available memory, even considering that it might take a significant amount of memory in cases that were not solved. This is because it implements specific rather conservative limits on memory usage. If any thread exceeds the allotted memory, ARVANDHERD either returns immediately or is restarted with a less greedy search, depending on the solver and the number of failures that occurred. At the other extreme, PHSFF tends to use less memory but also fails more often on memory; it failed on memory in the example shown in Fig. 19. This is because it does not implement any memory policies and relies only on its low memory consumption.

4.4.4. *Distinguished performers in the sequential multi-core track*

According to the results in Table 6 (page 101), ARVANDHERD and AYALSOPLAN solved more problems with the highest score. In particular, the difference between ARVANDHERD and all the other entrants, as evidenced by the statistical test performed in Section 4.4.1 for which results are summarized in Fig. 16 (page 103), is deemed statistically significant. AYALSOPLAN appears to be dominated by ROAMER-P and has indistinguishable performance from PHSFF. However, these two planners were ranked far behind AYALSOPLAN. Thus, the following planners were distinguished by their performance in the sequential multi-core track of IPC-2011:

- *Winner*: ARVANDHERD, with an overall score of 227.07 and 236 problems solved.
- *Runner-up*: AYALSOPLAN, with a final score of 159.95 and 184 problems solved.

Table 7
Score, number of problems solved, and success rate for the temporal satisficing planners.

	DAE_YAHSP	YAHSP2-MT	POPF2	YAHSP2	LMTD
Score	126.16	111.14	110.60	98.97	57.75
Solved	136	145	119	137	62
Success rate	56.67%	60.41%	49.58%	57.08%	25.83%
	CPT4	SHARAABI	TLP-GP		
Score	44.41	0	0		
Solved	46	0	0		
Success rate	19.16%	0%	0%		

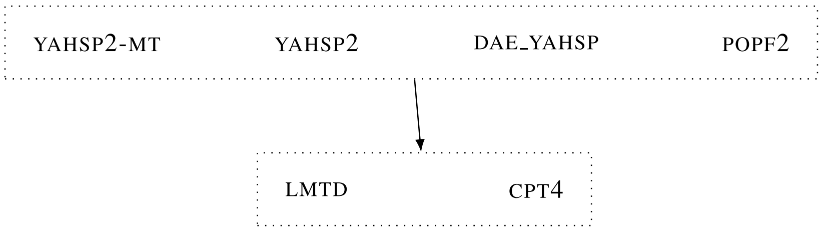


Fig. 22. Partial order for planner performance in the temporal satisficing track in terms of successfully solved problems according to the binomial test with $p = 0.5$. An alternative confidence level of 99.5% did not change the results.

4.5. *Performance of the temporal satisficing planners*

The following subsections examine the performance of entrants in the temporal satisficing track under the usual parameters: number of problems solved, quality, CPU time, and memory management.

4.5.1. *Number of problems solved and quality*

In this track, planners were required to find a valid solution to a planning task that involves durative actions that might temporarily overlap or interfere. A key difference for this track is that planners were not required to minimize the total cost of the actions in the plan, but the makespan. In fact, there is no notion of cost other than the duration of each action for the purpose of the competition, but of course costs can be defined in general for temporal planning.

It has already been noted that temporal planning can be computationally more complex than classical planning [10]. In fact, it was found that concurrency differentiates expressive temporal action languages from simple ones [27]. Nevertheless and despite the availability of a temporal extension to PDDL since 2003 [39], most domains in previous IPCs were inherently sequential so that concurrency was not required to solve the problems [40].[11]

In IPC-2011, three of 12 domains required concurrency explicitly: MATCHCELLAR (Appendix B.2.4), TMS (Appendix B.2.11) and TURNANDOPEN (Appendix B.2.12). Planners were allowed to produce multiple solutions and all of them were verified with VAL, which found that up to 40.1% of the plans issued were invalid, most of them produced by DAE_YAHSP, YAHSP2, and YAHSP2-MT in the domains that explicitly required concurrency.

Table 7 shows the final score, the number of problems solved, and the success rate for the temporal satisficing planners. In this case, the coefficient for correlation between the number of problems solved and the score is 0.981. The most remarkable observations are that YAHSP2-MT and YAHSP2 are the first- and second-ranked planners according to the number of problems solved, yet their score ranks them second and fourth, respectively. In this regard, almost no statistical dominance was found among the top-ranked planners, as discussed next. Remarkably, two planners did not solve a single problem: SHARAABI and TLP-GP. In particular, SHARAABI [41] actually solved 63 instances, but all the plan solutions were found to be invalid according to VAL. The problem was that SHARAABI did not shift the start time of actions consuming a resource immediately after the preceding action produced it; instead, one ended precisely at the same time as the other started. This is invalid according to VAL. For TLP-GP, the authors reported a bug in the parser that prevented it from ever solving a single instance.

Planners SHARAABI and TLP-GP were removed from the following statistical tests as they were not able to solve any problem. Fig. 22 shows the binomial test results for coverage. As anticipated, the statistical test distinguishes the performance of all planners with regard to coverage in just two levels, placing the four top-ranked planners in the first and the other two in the second. On one hand, while YAHSP2-MT is the planner providing the best coverage, its performance is very close to that of both DAE_YAHSP and YAHSP2. On the other hand, in spite of its differences in coverage from POPF2, both planners solve rather different sets of problems: DAE_YAHSP, YAHSP2-MT, and YAHSP2 are *epoch-based* planners, which gives them a remarkable advantage when solving simple temporal domains, but they are incomplete, as witnessed by their poor performance in concurrent domains, for which they solved no problem at all. POPF2 and LMTD were the only planners able to

[11] Although, as mentioned in Section 3.1, a number of them were introduced in the fourth IPC.

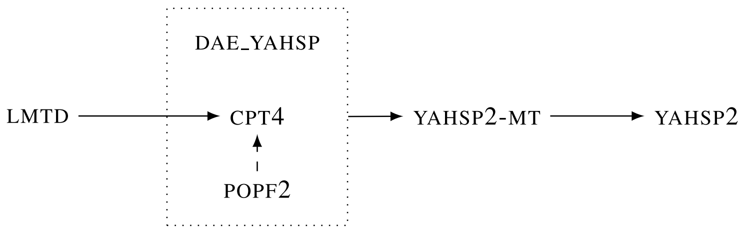


Fig. 23. Partial order for planner performance in the temporal satisficing track in terms of quality (or, equivalently, makespan) according to the Wilcoxon signed-rank test considering only double hits. An alternative confidence level of 99.5% did not change the results.

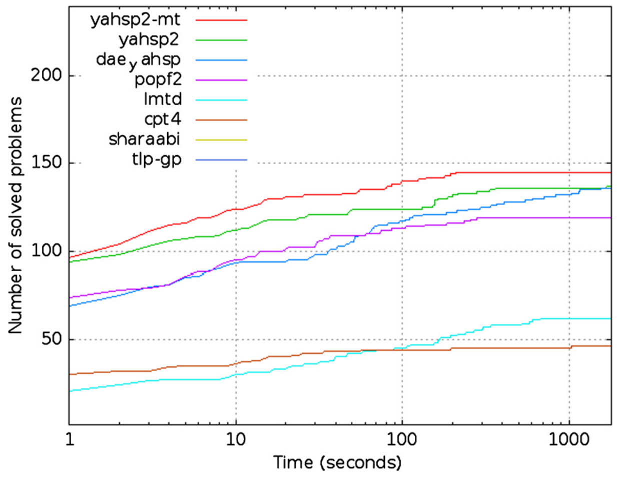


Fig. 24. Analysis of the evolution of the number of problems solved by planners in the temporal satisficing track over time in the range [1, 1800) s. The x-axis is shown on a logarithmic scale.

find solutions to domains that required concurrency. Indeed, POPF2 solved all the problems in the MATCHCELLAR domain, nine in TURNANDOPEN, and five in TMS. Its performance in these domains is only rivaled by LMTD, which solved 15 problems in MATCHCELLAR, 13 in TURNANDOPEN, and none in TMS. It seems reasonable to assume that if more domains requiring concurrency had been selected, the final scores would have changed substantially.

The situation becomes even more apparent when considering differences in quality or makespan for the solutions found by each planner. The Wilcoxon signed-rank results are shown in Fig. 23. CPT4 stands out, as expected, since it is an optimal planner. However, LMTD dominates CPT4 in terms of makespan, with medians for the total time distribution of 7 and 7.17, respectively. This is a counterintuitive result since LMTD is a satisficing temporal planning system, whereas CPT4 is an optimal planner, so this observation deserves an explanation. CPT4 is indeed an optimal planner (and this explains its low coverage) for both makespan and total cost (the latter was not tested in IPC-2011). However, when optimizing total time it actually produces optimal plans with regard to the conservative semantics [42] and not to PDDL 2.1 semantics, although the latter allow more concurrency and thus shorter makespans. Obviously, CPT4 could have easily performed a post-processing step to compress the plans, switching from conservative to PDDL 2.1 semantics, but CPT4 was never designed to produce such plans according to its author (Vincent Vidal, personal communication). LMTD uses a heuristic called *temporal precedence constraints contexts*, h^{tpcc} [43], whose efficiency strongly depends on the selection of preference constraints among preconditions. Although it is unclear whether this heuristic works in the general case or not, LMTD performed exceptionally well according to the Wilcoxon signed-rank test,[12] and this is attributed mainly to the accuracy of its estimates in those cases. To conclude, the median of the total time distribution for the plans generated by LMTD is very close to the medians of the same distributions generated by POPF2 and DAE_YAHSP, but is greater than those of YAHSP2-MT (9 versus 10.14) and YAHSP2 (9 versus 11.66). The latter differences are deemed statistically significant, even at a restrictive confidence level of 0.001.

4.5.2. *Analysis of CPU time*

Fig. 24 shows an overall view of the number of problems solved by each planner and how quickly they provided responses; the makespan of the plans generated by each planner is ignored. The raw speed performance of the planners falls into four different categories: (i) YAHSP2-MT dominates all the other entries; (ii) YAHSP2 dominates DAE_YAHSP and POPF2

[12] Recall that when comparing plan quality, only double hits are considered.

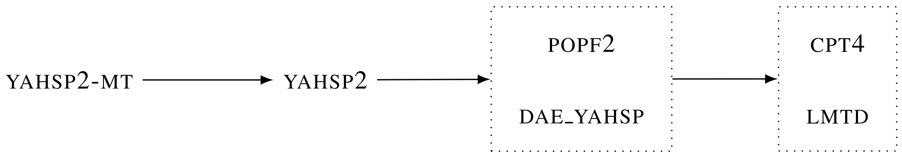


Fig. 25. Partial order for planner performance in the temporal satisficing track in terms of time taken to find the first solution according to the Wilcoxon signed-rank test. An alternative confidence level of 99.5% did not change the results.

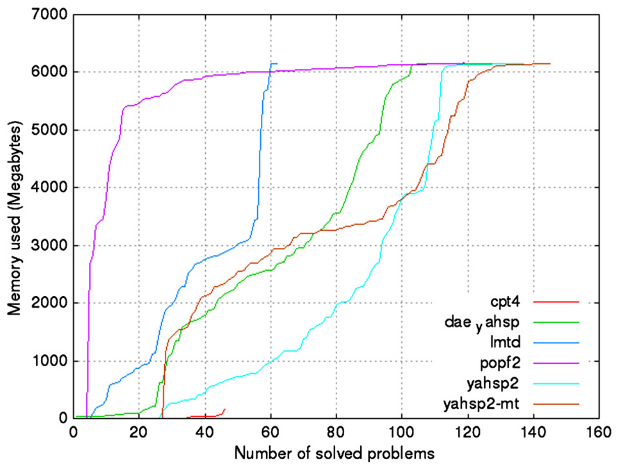


Fig. 26. Maximum memory (y-axis, MB) taken by each entry in the temporal satisficing track in solving a specific number of planning tasks.

most of the time, except at the end, where DAE_YAHSP solves one fewer problem than YAHSP2 does; (iii) CPT4 and LMTD perform similarly and are dominated throughout the interval by the previously mentioned planners; and (iv) SHARAABI and TLP-GP are included only for the sake of completeness, but their curve is hidden because they did not solve any problem. Interestingly, DAE_YAHSP progresses by solving problems throughout the interval, whereas all the other planners stagnate relatively soon. For example, YAHSP2-MT, the winner of the track, had solved 144 problems at $t = 196$ s and only solved one additional problem in the remaining 1604 s.

The Wilcoxon signed-rank results for raw speed are shown in Fig. 25. Again, SHARAABI and TLP-GP have been excluded from this analysis because they did not solve any problem. As noted above, YAHSP2-MT was the fastest algorithm providing responses, followed by YAHSP2. This observation is supported by Fig. 24, which indicates that these planners were the fastest throughout the interval. As expected, YAHSP2 was faster than the other planners in the second group: the winner of the track, DAE_YAHSP, and one of the joint runners-up, POPF2. The performance of these planners in terms of raw speed is significantly superior to that of planners in the third group: CPT4 and LMTD.

4.5.3. Analysis of memory performance

In the temporal satisficing track there were no portfolios or planners that use memory-based heuristics. Thus, the most usual policy for handling memory consisted of incremental use. Closer inspection of the memory consumption by all entrants in this track revealed that this was precisely the case. As an example, Fig. 27 (page 109) shows how much memory was used over time in solving problem 003 of the ELEVATORS domain, which was solved by all the planners except CPT4, SHARAABI, and TLP-GP. It is evident that memory was used linearly and increased over the whole period. Note, however, the flat profile for POPF2 from $t = 240$ s up to $t = 1000$ s. This results from the particular choice of algorithms by POPF2. From $t = 0$ s to $t = 240$ s, POPF2 uses the enforced hill-climbing search algorithm (EHC) [31], retaining the set of visited states to avoid visiting duplicate states. This set grows, as shown in Fig. 27. At $t = 240$ s, EHC stops searching and the set of visited states is cleared, although POPF2 retained the memory internally. From $t = 240$ s to $t = 1000$ s, a weighted-A* (WA*) search is used and a set of visited states is retained, which grows, filling the memory retained by POPF2. At $t = 1000$ s, the set of visited states from WA* has filled the memory retained, so memory usage starts to increase again from this point onwards.

In general, it was observed that POPF2 was very memory-demanding, using all of the available memory in most cases.[13] By contrast, CPT4 was very conservative, using less than 1 GB most of the time. Fig. 26 shows the maximum memory needed to solve a different number of problems.

[13] This was apparently due to an internal bug that was corrected after the competition (Andrew Coles, personal communication). In other cases, POPF2 failed to solve problems (e.g., it solved no problem at all in the FLOORTILE domain) because of a parsing issue that was also sorted out after the competition (Amanda Coles, personal communication).

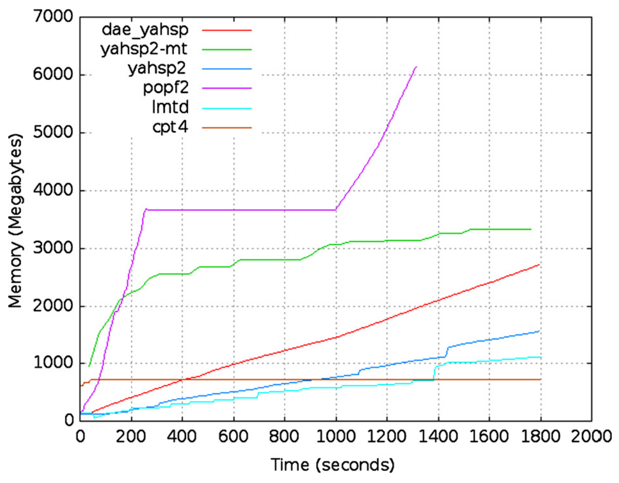


Fig. 27. Examples of typical memory profiles observed in the temporal satisficing track. The figure shows the memory profile for all entrants except SHARAABI and TLP-GP in solving problem 003 of the ELEVATORS domain.

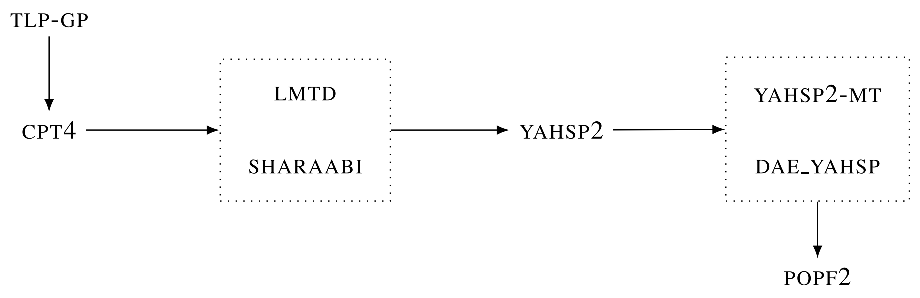


Fig. 28. Partial order for planner performance of planners with regard to memory usage in the temporal satisficing track according to the Wilcoxon signed-rank test. An alternative confidence level equal to 99.5% did not change the results.

Fig. 28 shows the acyclic graph representing the statistical dominance among all entrants in terms of memory consumption according to the Wilcoxon signed-rank test. The results are not surprising and the only remarkable observations seem to be that TLP-GP uses even less memory than CPT4 and that POPF2, as anticipated, uses more memory than all the other planners.

4.5.4. Distinguished performers in the temporal satisficing track

According to the results in Table 7 (page 106), the planners DAE_YAHSP and YAHSP2-MT have the highest score. However, the following points are noted:

- The difference in score between YAHSP2-MT and POPF2 is very small (about half a point), with the former awarded more points than the latter. However, the statistical test conducted in Section 4.5.1 (Fig. 23, page 107) concluded that the latter tends to produce plans with better quality than the former does.
- Table 7 (page 106) reports that YAHSP2 solved an additional problem over DAE_YAHSP. However when considering quality, this planner is statistically dominated by both DAE_YAHSP and YAHSP2-MT. Since the IPC scoring schema emphasizes quality, this planner was discarded among the list of top performers.

Thus, the following planners were distinguished by their performance in the temporal satisficing track of IPC-2011:

- *Winner*: DAE_YAHSP, with a total score of 126.16 and 136 problems solved.
- *Runner-up*: In view of the preceding considerations, the following planners were named joint runners-up:
  - YAHSP2-MT, with 111.14 points and 145 problems solved, and
  - POPF2, with 110.60 points and 119 problems solved, and distinguished performance in domains requiring concurrency.

5. Scalability analysis

In this section, two different questions related to *scalability* are analyzed separately. In the first subsection, we follow the fifth IPC practice [38] and compare the performance of the winners of IPC-2011 and IPC-2008 for the same benchmarking domains. In the second subsection, we compare the performance of the winners of the sequential satisficing and multi-core tracks to determine whether the availability of more computational resources resulted in either more coverage or better plan quality.

5.1. How good is the performance of the IPC-2011 planners?

We now compare the performance of the winners of IPC-2011 with the winners of IPC-2008 for the same planning tasks: the problems selected for IPC-2011. If refined versions of the winners of IPC-2008 were available, these were preferred over the versions submitted at the time, which gives an additional advantage to those entrants. This happened in the sequential satisficing and optimal tracks, for which the versions entered in IPC-2011 were used instead of the old versions. This analysis was not possible for the sequential multi-core track since it was run for the first time in IPC-2011.

5.1.1. Sequential optimal track

The winner of the sequential optimal track in IPC-2008 was GAMER. A new version of this planner was entered in IPC-2011 and therefore it was selected for comparison with the winner of IPC-2011, FDSS-1. The comparisons are performed with regard to coverage and raw speed. According to the analysis in this section, overall, the optimal planner that won IPC-2011 improved on the performance of the optimal planner that won IPC-2008 with regard to the problems chosen for IPC-2011 in terms of both coverage and raw speed.

Table 4 (page 92) shows that FDSS-1 solved 185 problems and GAMER solved 37 fewer. The difference in coverage is 13.22% over the whole set of planning tasks, but an improvement of 20% for the number of problems solved by the IPC-2008 winner. In addition, 136 problems were solved by both the planners. This leaves 12 problems that were solved by GAMER but not by FDSS-1, and about four times more problems solved by the latter that could not be solved by the former. These data, in conjunction with the observation that GAMER ranked third from the bottom, suggest a significant improvement over the current state of the art before the competition.

Fig. 3 (page 93) shows that the evolution of coverage over time for GAMER is dominated by almost every participant in IPC-2011. Again, this suggests a significant improvement over the previous state of the art.

Finally, the organizers of IPC-2008 reported that GAMER was marginally surpassed at that time by the baseline planner, a best-first search with a blind heuristic denoted as BLIND. While GAMER solved 115 problems in 2008, BLIND managed to solve an additional instance. Therefore, the same baseline planner was compared with the performance of the newest version of GAMER and the winner of IPC-2011, FDSS-1. The differences in coverage are deemed as definitive: BLIND solved 119 instances from the benchmarking set of IPC-2011, while the newest version of GAMER and FDSS-1 solved 148 and 185, respectively, as mentioned above. Again, this result suggests a significant improvement in the current state of the art, in optimal planning at least, with regard to the benchmarking set chosen in both IPC-2008 and IPC-2011.

5.1.2. Sequential satisficing track

In 2008 the winner of the sequential satisficing track was LAMA-2008 and a refined version of this planner was entered in IPC-2011. Thus, this version was compared with the current winner, LAMA-2011, instead of the version submitted in 2008. Again, the analysis in this section shows that, overall, the planner that won IPC-2011 improves on the performance of the IPC-2008 winner for the problems chosen for IPC-2011 in terms of coverage, quality, and raw speed.

As shown in Table 5 (page 96), LAMA-2008 solved 188 problems and LAMA-2011 solved 250. This represents an improvement of almost 25% in coverage. Not surprisingly, LAMA-2011 solved all the problems solved by LAMA-2008 apart from just five. Interestingly, other planners not directly based on the same techniques used by LAMA-2008, such as FORKUNIFORM and PROBE, solved significantly more problems.

Regarding CPU time, Figs. 10(a) and 10(b) (page 99) show the evolution of coverage over time for the 12 top-ranked planners. Clearly, LAMA-2008 is dominated by a number of participants, including LAMA-2011, as evidenced by the Wilcoxon signed-rank test (Fig. 11, page 100).

5.1.3. Temporal satisficing track

The winner of the temporal satisficing track in 2008 was SGPLAN6. Since it was not entered in IPC-2011, the same version submitted in 2008 was used. Strikingly, SGPLAN6 was surpassed by the baseline solver used in IPC-2008, which consisted of the FF planner [31] after dropping the temporal definitions and scheduling the resulting plan using the critical path algorithm. Therefore, the baseline planner (denoted here as BASE) was also used for comparison with the IPC-2011 winners. The analysis in this section shows that, overall, the temporal planner that won IPC-2011 improves on the performance of the temporal planner that won IPC-2008 for the problems chosen for IPC-2011 in terms of quality, but not necessarily coverage, and that it performs worse in terms of raw speed. The analysis also highlights the importance of benchmarking domain selection, and we report here that, contrary to the conclusions reached for IPC-2008, BASE is not necessarily better than SGPLAN6.

Regarding coverage, SGPLAN6 effectively solved 142 problems, six more than the winner of IPC-2011, DAE_YAHSP, and only three fewer than the planner with the best coverage, YAHSP2-MT (Table 7, page 106). However, contrary to the observations made for IPC-2008, BASE solved 116 problems, which represents a decrease of almost 20%. To confirm these observations, a binomial test on coverage was conducted in a separate track for which the only participants were DAE_YAHSP, POPF2, YAHSP2, YAHSP2-MT, SGPLAN6, and BASE. The resulting p-values (shown in parentheses) indicate that BASE performed significantly worse in terms of coverage than YAHSP2-MT (2.85×10^{-5}) at a restricted confidence level of $\alpha = 0.001$. The same hypothesis is accepted at a confidence level of $\alpha = 0.005$ with respect to YAHSP2 (1.2×10^{-3}) and SGPLAN6 (1.4×10^{-3}). However, the differences in coverage compared to DAE_YAHSP (7.5×10^{-3}) and POPF2 (0.41) were not statistically significant. Comparison of coverage against SGPLAN6 revealed no statistically significant dominance.

The scenario is more illustrative if the official metric of the competition is applied to this separate track (planner score shown in parentheses): the winner and two runners-up for IPC-2011, DAE_YAHSP (122.32), YAHSP2-MT (110.52), and POPF2 (110.51), are still ranked first, second, and third, respectively, with SGPLAN6 (101.23) ranked fourth and BASE ranked last with a significantly lower score of 92.07 after YAHSP2, which was awarded 98.66 points. On combining these results with the observation that SGPLAN6 showed remarkable coverage, it is easy to conclude that SGPLAN6 generates plans of lower quality than the distinguished planners in IPC-2011. To test this conjecture, the Wilcoxon signed-rank test was performed for quality, which revealed that SGPLAN6 does generate worse plans (i.e., with larger makespan) than DAE_YAHSP and POPF2 at a confidence level of $\alpha = 0.001$, with p-values of 1.22×10^{-9} and 1.11×10^{-16}, respectively. There is significant evidence that SGPLAN6 generates plans worse than those of YAHSP2-MT at a confidence level of $\alpha = 0.005$, with $p = 3.29 \times 10^{-3}$.

This difference in performance between BASE and SGPLAN6 for a different set of planning tasks deserves an explanation. In IPC-2011, six domains were reused from IPC-2008: CrewPlanning, Elevators, Openstacks, ParcPrinter, PegSolitaire, and Sokoban. For this set of planning tasks, BASE solved 93 problems and SGPLAN6 solved 82. For the new domains introduced in IPC-2011, both planners failed to solve any of the planning tasks requiring concurrency (in MatchCellar, TMS, and TURNANDOPEN). Both planners solved all problems in the Parking domain. However, SGPLAN6 exhibited far superior performance to BASE in the last two domains, Storage (reused from IPC-2006) and Floortile (introduced for the first time in 2011), for which it solved all the problems, whereas BASE solved only three problems in the Storage domain and none in the Floortile domain. The overall difference between the planners when comparing their performance in 2008 and 2011 is attributed solely to the selection of benchmarking problems.

Finally, a Wilcoxon signed-rank test was conducted for CPU time required to find the first solution (p-values in parentheses): SGPLAN6 was much faster than DAE_YAHSP (8.23×10^{-9}) and POPF2 (3.68×10^{-5}), but was not significantly faster than YAHSP2 (0.16) or YAHSP2-MT (0.19).

5.2. How good is the performance of the sequential multi-core planners?

Multiple cores have been a standard CPU feature for at least 5 years at the time of writing. Although this availability might have fostered the development of multi-core planning systems, participation in the sequential multi-core track was much lower than in its single-core counterpart, the sequential satisficing track. However, "*the relation between a [...] multi-core track and the corresponding single-core track requires special attention, and the results of the tracks cannot really be considered in separation from each other*" (Jussi Rintanen by e-mail, March 14, 2013).

As mentioned in Section 3.2, the entrants in both tracks were intentionally given the same planning tasks for ease of comparison. Here we compare the performance of the winners of both tracks with regard to coverage and plan quality. Tables 5 (page 96) and 6 (page 101) show that LAMA-2011 and ARVANDHERD solved 250 and 236 problems, respectively. Thus, the winner of the sequential satisficing track showed better coverage despite using only a single dual core instead of four dual cores.

Unfortunately, it is not as straightforward to compare the plan quality for these two planners because the scoring schema used in IPC-2011 may assign a planner a better score simply because the other planners in the same track were worse (Sections 2.3 and 6). Thus, to provide a better assessment of differences in plan quality between LAMA-2011 and ARVANDHERD, an alternative competition was simulated with all entrants in both the sequential multi-core and satisficing tracks, yielding a total number of 35 entrants. In this way, both planners were compared with regard to the solutions provided by the same set of planners. The results indicate that the score for LAMA-2011 slightly decreased from 216.33 to 215.51, while the score for ARVANDHERD decreased even more, from 227.07 to 210.41. The most prominent differences were observed for the NOMYSTERY and PARKING domains; ARVANDHERD showed superior performance in the former but worse performance in the latter. Overall, LAMA-2011 would have been declared the winner and ARVANDHERD would have been selected as the best runner-up; its difference from the third-ranked planner, FDSS-1, in this simulated competition is even greater, since FDSS-1 was assigned 201.47 points. In the final ranking, up to 10 classical planners obtain a better score than the second-ranked multi-core planner, which shows clear superiority of single-core over multi-core planners at the time of writing.

Note that these analyses were performed using the same time bounds. This observation is definitely relevant: "*Assuming you have a good handle on how to distribute the workload [...], multi-core planning essentially means increasing the runtime [...] bound. So the question is if, e.g., you can find significantly better plans in 4 h vs. 30 min if you compare an 8-core planner to a 1-core planner with a timeout of 30 min*" (Malte Helmert by e-mail, March 14, 2013). Despite some differences in how time is treated in both tracks (planners in the sequential multi-core track were given 30 min of wall-clock time, while the sequential satisficing planners were given 30 min of CPU time), the observation is valid even if it seems unrealistic that sequential multi-core planners could achieve linear increases in speed. From the results outlined above, it is clear that state-of-the-art multi-core planners perform worse than single-core satisficing planners for the same time bounds, needless to mention they do if the satisficing planners are given more time proportionally to the number of available cores.

The fact that the winner of the sequential satisficing track surpassed the performance of the winner of the sequential multi-core track (at least in terms of both coverage and the official IPC score, and in terms of plan quality when considering the tasks solved by both planners) is not of great concern. The history of the IPC shows that classical planners are highly engineered in terms of data structures and are difficult to beat in the first editions of new tracks.

6. Discussion of the scoring schema

This section critically analyzes the scoring schema used at IPC-2011, considers a number of alternatives, and discusses whether the results might have been different. However, it should be noted that the participants were aware of the scoring schema well ahead of the submission deadline, so it is reasonable to assume that their strategies were somehow tailored towards it. The following discussion pursues two goals: (i) a study of the robustness of the official results; and (ii) the provision of additional data on the impact of different metrics that might serve as a criticism of the results of IPC-2011 and an inspiration for new measures in future competitions. A preliminary discussion of the scoring schema has been already presented [44]. This section reviews the concepts presented there but the most important contribution is how they lead to the definition of alternative criteria and the comparison in practice with those alternatives.

Although the scoring schema (Section 2.3) primarily serves to declare a winner and a runner-up, it should faithfully reflect the performance of all planners to be effective. We make the following observations.

Observation I. The official scoring schema is not linear and tends to favor planners that effectively solve instances with low optimal cost, provided that other planners deviate from it, even if only slightly [44]. Consider two planners A and B for four planning tasks. Planner A finds solutions with a total cost equal to 1, 2, 10, and 11, whereas planner B generates plans of quality 2, 4, 6, and 8. The second planner tends to create plans that are shorter and, indeed, the median and mean of its cost distribution equal 5, while the mean and median of the first planner equal 6. Assuming that the optimal costs are 1, 2, 6, and 8, respectively, planner A would be awarded $\frac{1}{1}+\frac{2}{2}+\frac{6}{10}+\frac{8}{11}=3.32$, whereas planner B would get $\frac{1}{2}+\frac{2}{4}+\frac{6}{6}+\frac{8}{8}=3$, which is less than 3.32, in contrast to the expected outcome. From this example it is clear that the scoring schema is more sensitive to differences when the optimal solutions are shorter than when they are larger.

Observation II. The official score regards all problems as being equally important, but this is not necessarily true. A clear example of this inefficiency is given in Section 4.5.1, in which planning tasks requiring concurrency were finally taken into account to provide an additional award beyond the ranking computed according to the official score.

Observation III. The official score requires that the optimal solutions are known beforehand, and these can be difficult to derive in some cases. It is straightforward to prove that the error produced when not using the optimal solutions as a reference is equal to the quotient C_{best}/C^*, where C_{best} is the cost of the best plan found and C^* is the optimal cost. To prove this, observe that the score for planner p when using the best solution found as a reference for a specific planning task i is

$$S_i^p = \frac{C_{best}}{(C_{best}+\delta_i^p)},$$

where δ_i^p is the excess committed by planner p over the best solution found (and necessarily equal to zero for at least one planner) when solving instance i. That is, $(C_{best}+\delta_i^p)$ denotes the cost of the plan generated by planner p for planning instance i. The theoretical score for planner p should be computed as

$$S_i^{p^*} = \frac{C^*}{(C_{best}+\delta_i^p)}.$$

From the preceding expressions, we have

$$\frac{S_i^p}{S_i^{p^*}} = \frac{C_{best}}{C^*},$$

so that planner p is assigned a score that is affected by a factor equal to the ratio of the cost of the shorter plan found to the cost of the optimal solution. Since $C_{best} \geq C^*$ by definition, not using the optimal solutions as a reference increases the score for all planners. This observation has two immediate consequences.

- Although all planners see their score affected by the same constant, this value can be different for different planning tasks and can induce a final ranking that differs from the theoretical one.
  For clarity, assume that two planners, A and B, solve two different tasks. Assume further that A solves both tasks with total costs of 100 and 110, respectively, and B finds solutions with total costs 120 and 100. The best solution found is 100 for both planning tasks, so that the score for A is $s_A = \frac{100}{100}+\frac{100}{110}=1.909$. Likewise, B is awarded $s_B = \frac{100}{120}+\frac{100}{100}=$ 1.8333 points, so the first planner would be declared the winner. Assume, however, that both tasks can be optimally

solved with action costs equal to 20 and 50, respectively. In this case, the score for each planner for each planning task decreases by a factor equal to 100/20 and 100/50, respectively. Thus, the theoretical score for the first planner is $s^*_{\mathrm{A}} = \frac{20}{100} + \frac{50}{110} = 0.6545$ and the second planner should be awarded $s^*_{\mathrm{B}} = \frac{20}{120} + \frac{50}{100} = 0.6666$ points. The second planner performs better (although marginally in this example) than the first when considering the cost of the optimal solutions. From a fair evaluation perspective, this is problematic, since it can violate the decision-theoretic axiom of the *independence of irrelevant alternatives* [44].
Assume that the outcome of a planning competition is such that P_1 wins by narrowly outscoring P_2. Assume further that the same competition is repeated, but this time with an additional planner P_3 that is clearly worse overall compared to P_1 or P_2, but improves over the best known solutions for a few problems. This can affect the balance between P_1 and P_2 in such a way that P_2 becomes the winner. In other words, the additional participant P_3 influences which of P_1 and P_2 is considered the best planner. This type of situation is highly undesirable since it can introduce strategic considerations into the competition that run counter to its scientific goals. For example, a team of researchers might refrain from entering planner P into the competition because this might adversely affect the performance of another planner Q entered by the same team of researchers.
A good way to avoid or reduce such quandaries is to use strong domain-dependent solvers to find high-quality (or even optimal) reference plans, so that the best known solutions will not be ones (uniquely) found by the participating planners. Many such specially developed domain-dependent solvers, and in some cases human-generated solutions, were used at IPC-2008 for this reason. However, no optimal solver was developed for IPC-2011 [44].

- An unwanted effect of not using optimal solutions as a reference baseline is that the final score is not properly scaled over the optimal performance, but instead over the best performance observed.
In the preceding example, the scores $s_{\mathrm{A}} = 1.909$ and $s_{\mathrm{B}} = 1.8333$ are too close to the maximum achievable score of 2, which suggests that both planners perform very well but we can only say that they perform similarly. However, the scores obtained according to the optimal solutions, $s^*_{\mathrm{A}} = 0.6545$ and $s^*_{\mathrm{B}} = 0.6666$, have a more meaningful interpretation: both planners, with similar performance, are far from the optimal solutions.

In the following analysis, planners are ranked according to an additional number of metrics. To justify the various alternatives, we note that while IPC-2011 had an emphasis on plan quality, other metrics can be considered for different needs.

- *Coverage*: this function awards one point to each planner that solves the current task and zero otherwise.
This metric is not affected by Observation I since all planners are awarded either one point or none per planning task regardless of their computation effort or that of competitors. Likewise, the metric is not affected by Observation III: even if we assume that all planning tasks are solvable, the ratio C_{best}/C^*[14] equals either one (if at least one planner finds a solution) or zero (if a planning task is not solved by any entrant), so that the score for all planners is never affected. However, it does not address Observation II, since all planning tasks are considered to be equal.
- *Time*: this function computes the score for a planner for a given task as the quotient
$$\frac{1}{1+\log(\frac{T}{T^*})},$$
where T is the time taken by the planner to solve a particular planning task and T^* is the minimum time required by any planner to solve the same task. Since time differences can become arbitrarily large, logarithms are used for scaling time scores properly. In this metric, times of less than 1 s are considered to be negligible and are normalized to be equal to 1. This was the official metric for evaluating time performance in the learning track in IPC-2008.
This metric makes an explicit effort to address Observation I by taking a logarithm of the ratio that represents the speed of each planner in relation to the fastest one. However, with regard to Observation II it considers all planning instances to be equally important, neglecting the importance of solving the tasks that were found to be particularly hard by other planners. Finally, Observation III is irrelevant since there is no notion of an optimal solution here and comparison with the best observed performance makes sense.
- *QT*: this function computes for each planner and task a tuple (Q, T), where Q denotes the quality of the best solution found by a given planner and T is the time taken by that planner to find it. If no solution is found, constant values that represent infinitely bad quality and time are assigned. Next, this metric awards each planner a score that equals the quotient of the number of distinct tuples a planner Pareto-dominates and the maximum number of distinct tuples Pareto-dominated by any planner with regard to the same planning task. In the current context, (Q, T) is said to Pareto-dominate (Q', T') if and only if $Q \geq Q'$ and $T < T'$.
The normalization for this metric results in a relatively small penalty for planners that Pareto-dominate only a few tuples if the maximum number of dominances is small. This is why only the number of distinct tuples is considered when computing the Pareto dominance, so that the effect of Observation I is diminished. However, the metric explicitly

[14] In this particular case, C_{best} should be read as "1 if this planning task is solved by any entrant and zero otherwise"; similarly, we assume that C^* is always equal to 1, since all planning tasks are solvable by hypothesis.

Table 8
Score for all entrants in the sequential optimal track with regard to the following metrics: *Coverage*, *Time*, and *QT*.

	FDSS-1	FDSS-2	SELMAX	M&S	LMCUT	FD-AUTOTUNE
Coverage	185	182	169	169	167	166
Time	148.67	146.79	125.60	140.57	130.47	127.43
QT	156	153	140	140	138	137
	FORKINIT	BJOLP	LMFORK	GAMER	IFORKINIT	CPT4
Coverage	158	151	148	148	144	44
Time	103.33	114.76	79.09	76.61	110.40	36.49
QT	129	122	119	119	115	15

addresses Observation II by making comparisons with the performance of all the other planners one by one, instead of comparison with the best performer per instance. However, it does not address Observation III, since it uses the best observed quality instead of the optimal quality.[15]

Recall from the different subsections devoted to each track in Section 4 that the coefficient for correlation between the official competition metric and *Coverage* was very close to 1. To provide additional evidence of the correlation between these metrics, the Spearman rank-order correlation test was used (Appendix E). Detailed data on the results for these metrics for all planners and domains are available in the RESULTS section of the competition website[16] for all tracks. The following analysis provide just an overview of the main findings.

Table 8 lists the score for all entrants in the sequential optimal track according to the metrics *Coverage*, *Time* and *QT*; *Score* has been omitted because it is equal to *Coverage*. The first observation is that the *QT* score is perfectly correlated with the *Coverage* score: in this track, all plans have strictly the same quality and therefore Pareto dominance is not feasible unless one planner effectively solves a problem and at least another one does not. In spite of the number of planners that did not solve a particular task, the tuple that represents their performance is unique and is symbolized by arbitrarily large constants. Thus, the number of distinct Pareto-dominated tuples is always equal to 1. The score given by the metric *QT* equals the *Coverage* minus the number of instances simultaneously solved by all entrants, for which no Pareto dominance is then possible. From Table 8 it is evident that the difference between *Coverage* and *QT* is always 29. Indeed, precisely 29 problems were solved by all entrants in this track (the particular problems are indicated in parentheses with ranges shown by an *en dash*): NOMYSTERY (000–003, 010–014), PARCPRINTER (000–004, 007–008), PEGSOL (004), SCANALYZER (000), VISITALL (002–007, 009, 011) and WOODWORKING (000–002).

The Spearman rank-order correlation coefficient for *Coverage* and *Time* is 0.912, which is statistically significant at $\alpha = 0.01$, so that the alternative hypothesis that the metrics are correlated is accepted. Owing to the perfect correlation between *Coverage* and *QT*, comparison of *QT* and *Time* yielded exactly the same results. As a consequence of the strong (and sometimes even perfect) correlation among all the metrics, it can be concluded that the planners distinguished in this track perform equally well under the different metrics.

Table 9 lists the final scores for all entrants in the sequential satisficing track. The Spearman rank-order correlation results indicate that all the metrics are very strongly correlated at a confidence level of greater than or equal to 99.0%.

Table 10 shows the score for all entrants in the sequential multi-core track for all the metrics considered. According to the Spearman rank-order correlation test, the correlation between *Coverage* and *QT* is perfect, and these metrics rank the planners in precisely the same order. Owing to this perfect matching, the correlation between *Score* and *Coverage* is exactly the same as that between *Score* and *QT*, which is almost perfect (or even perfect in the Cohen classification [45]) with a correlation coefficient of 0.976, which is significant at $\alpha = 0.01$. Almost the same observation holds for the relationship between *Time* and either *Coverage* or *QT*, which are strongly correlated at $\alpha = 0.01$ (since $p = 0.007 < 0.01$) with a correlation coefficient equal of 0.857. However, as anticipated by the previous observations, the ranking according to *Time* is not the same as the official ranking. Nevertheless, the two-tailed significance for this statistical test regards them as being strongly correlated at $\alpha = 0.05$.

Table 11 (page 115) summarizes the final scores for all entries in the temporal satisficing track for the metrics considered in this section. This is the only case for which the track winner, DAE_YAHSP, greatly benefited from the metric selection. In fact, it ranked first only for the official metric, *Score*, and ranked second for the metrics *Coverage* and *QT*, and fourth for *Time*. By contrast, one of the two joint runners-up, YAHSP2-MT, was disadvantaged by this selection: although it was ranked second for *Score*, it was ranked first according to all the other metrics. The case of the other runner-up, POPF2, is untypical in the sense that although it was ranked either third or fourth for the different metrics, it was awarded runner-up position for its exceptional performance in domains that required concurrency, which is not necessarily reflected in the metric selection unless there is a significant number of domains of this type (in IPC-2011, only three out of 12 domains required concurrency).

[15] An interesting remark here is that addition of a fictitious tuple (Q^*, T^*), which denotes the cost of the optimal solution, does not work either, since this would penalize the planners that yield optimal solutions. This occurs because the spirit of Observation III emphasizes that we ignore whether one solution is optimal or not.

[16] http://www.plg.inf.uc3m.es/ipc2011-deterministic/Results.

Table 9
Score for all entrants in the sequential satisficing track with regard to the following metrics: the official metric (*Score*), *Coverage*, *Time*, and *QT*.

	LAMA-2011	FDSS-1	FDSS-2	FD-AUTOTUNE-1
Score	216.33	202.08	196.00	185.09
Coverage	250	232	233	223
Time	155.27	99.63	137.26	129.59
QT	207.98	163.73	180.79	172.65

	ROAMER	FD-AUTOTUNE-2	FORKUNIFORM	PROBE
Score	181.47	178.15	177.91	177.14
Coverage	213	193	207	233
Time	118.81	103.84	113.67	154.74
QT	170.38	151.96	158.11	185.35

	ARVAND	LAMA-2008	LAMAR	RANDWARD
Score	165.07	163.33	159.20	141.43
Coverage	190	188	195	184
Time	77.46	101.76	115.68	102.40
QT	137.74	151.98	150.55	138.16

	BRT	DAEYAHSP	CBP2	YAHSP2
Score	116.01	101.83	98.34	94.97
Coverage	157	120	135	138
Time	74.38	39.69	59.92	99.48
QT	115.83	74.82	90.18	96.35

	YAHSP2-MT	CBP	LPRPGP	MADAGASCAR-P
Score	94.14	85.43	67.07	65.93
Coverage	137	123	118	88
Time	97.07	56.84	72.68	77.91
QT	95.51	81.24	72.17	65.37

	POPF2	MADAGASCAR	CPT4	SATPLANLM-C
Score	59.88	51.98	47.85	29.96
Coverage	81	67	52	32
Time	41.93	48.52	32.41	16.58
QT	50.36	48.56	42.49	21.78

	SHARAABI	ACOPLAN	ACOPLAN2
Score	20.52	19.33	19.09
Coverage	33	20	20
Time	13.91	9.05	8.12
QT	17.71	12.58	12.42

Table 10
Score for all entrants in the sequential multi-core track with regard to the following metrics: the official metric (*Score*), *Coverage*, *Time*, and *QT*.

	ARVANDHERD	AYALSOPLAN	PHSFF	ROAMER-P
Score	227.07	159.95	130.59	129.06
Coverage	236	184	163	140
Time	131.65	94.63	154.99	54.69
QT	209.94	135.62	132.18	96.35

	YAHSP2-MT	MADAGASCAR-P	MADAGASCAR	ACOPLAN
Score	66.44	52.00	17.62	118.58
Coverage	88	67	18	153
Time	74.99	49.92	9.38	110.48
QT	59.57	39.47	9.40	115.06

Table 11
Score for all entrants in the temporal satisficing track with regard to the following metrics: the official metric (*Score*), *Coverage*, *Time*, and *QT*.

	DAE_YAHSP	YAHSP2-MT	POPF2	YAHSP2	LMTD
Score	126.16	111.14	110.60	98.97	57.75
Coverage	136	145	119	137	62
Time	71.65	136.23	89.31	124.69	36.15
QT	120.89	134.62	112.17	121.80	57.68

	CPT4	SHARAABI	TLP-GP
Score	44.41	0	0
Coverage	46	0	0
Time	43.48	0	0
QT	42.38	0	0

Assessment of these metrics in the sequential multi-core track revealed perfect correlation between *Coverage* and *QT* according to the Spearman rank-order correlation test. From this equivalence, it follows that *Score* and *Time* are equally correlated with *Coverage* and *QT*. All these correlations were statistically significant at a confidence level of $\alpha = 0.01$, but the correlation between *Score* and *Time* is accepted at a less restrictive confidence level of $\alpha = 0.05$.

7. Conclusions

Although most of the IPC organizational efforts are carried out by the competition organizers, the IPC series is a collaborative work. Our feeling is that this is one of the most (if not the most) important factors that explain its success. Other factors include its fostering of the development of new planners and its setting of deadlines for their completion. The IPC also contributes a number of planners to the public domain, as well as new benchmarking sets for performing new experiments. Overall, the IPC improves our understanding in a few specific ways.

However, the competition might also have some undesirable effects arising from the assumption that the IPC portrays the current state of the art in automated planning. Our view is that although IPC-2011 evaluated the largest number of planners for the largest number of domains ever, the competition results cannot be taken as an analysis of the current state of the art. Our analysis only reflects the performance of the participant planners in a subset of planning tasks under a particular evaluation set-up. Interesting planning tasks such as timeline-based planning, continuous planning, and real-time approximate reasoning were not included in the competition, and other useful evaluation set-ups, such as planning with small time bounds and exploiting further computational resources such as GPUs, were ignored. Comparisons among the same planners performed with different criteria might yield different results. This observation refers not only to formulae such as those discussed in Section 6 but also to the goal of the competition itself; for example, disproving plan existence has never been addressed in the IPC series [46,47]. Moreover, the benchmark domains do not cover the full range of possible planning tasks and the number of realistic domains is still low. In addition, it is very hard to come up with an unbiased selection of planning domains that does not favor any paradigm in particular. All of this should be carefully taken into account since one of the main dangers of the IPC being understood as a study of the current state of the art is that the community might concentrate research too much around the few issues usually considered in the IPC and the successful methods distinguished in the most recent IPC edition. In other words, the IPC might induce some form of standardization that would make "*our innovation to become boring and predictable*" (anonymous comment in a poll conducted among the planning community, October 2011).

As mentioned above, another advantage of the IPC is that it makes apparent the importance of contributing with standardized tools for automating the experimentation,[17] or creating public repositories for storing the best known results, among others. Our impression is that work should move away from organization of the IPC to focus on assisting in daily research. A highly desirable situation would be to run the IPC with the same or similar software that researchers use to conduct their own experiments. Again, this is not without risks and in the current context, standardized tools should be understood as readily available software for performing in the IPC the same (or similar) tasks that are performed by the research community. In this regard, tools can be useful, but distributors should be "*begged to put a bold and prominent disclaimer, explaining that these tools are not a replacement for analysis, effectively reminding researchers to perform their due diligence*" (anonymous comment in a poll conducted among the planning community, October 2011).

The IPC series represents a good opportunity to perform large-scale experiments from which it is possible, if not to draw definitive conclusions, at least to derive data for some questions that are often controversial because they might bias the competition. Let us discuss just a few that we feel are important. One of these is memory management. Automated planning systems are nothing more than computing systems. Thus, we extended the concern about the time bound to the memory bound and for the first time obtained data on memory consumption. Our recommendation, from a general point of view, is to offer as much memory as possible within reasonable limits (affordable memory in typical architectures at the time of organizing a new competition). Again, this prevents experimentation and selection of the best adapted planners to small devices such as mobile phones and tablets, which are already very common.

To shed some light on the most controversial issues, we carefully examined the scoring schema of the last two IPCs. There have been several criticisms of the scoring schema itself and the way it was applied in IPC-2008, and a few alternatives have been considered. Although different metrics might produce different rankings (as observed in the temporal satisficing track), strong correlations were found across all metrics, although it should be noted that *Score* and *Time* generally exhibited lower correlation. As noted elsewhere, this could result from the evaluation set-up for the competition, announced well ahead of the submission deadline. We have two recommendations for this issue: (i) research should be dedicated to this question to gain new insights; and (ii) a repository should be set up in which the community could submit and query best known solutions for a number of planning tasks to improve the evaluation of every planner with regard to the same planning tasks [48].

Another interesting question often regarded as controversial is selection of the benchmarking problems. We contributed here with specific means for automatically reusing problems from previous competitions. In the absence of a better un-

[17] There are already two good initiatives: LAB https://lab.readthedocs.org/ and the software used for IPC-2011 http://www.plg.inf.uc3m.es/ipc2011-deterministic/FrontPage/Software.

derstanding of the difficulty of planning tasks, a measure such as the Glicko rating system (Section 3.2 and Appendix D) might produce rankings of this difficulty. The key observation here is that problems can be compared with regard to their *performance* against planners. This is not without problems and generalizing this idea poses some interesting challenges. For example, maintaining a global list of the Glicko score for every planner might be difficult, because different versions of the same planner might easily proliferate, making it hard to trace the results and to compute the final score consistently across all versions. Likewise, there is another issue related to credibility, and the only results to be taken into account for updating the Glicko score should be from official events such as the IPC series.

From a technical point of view, IPC-2011 also contributed by introducing a brand new track, the sequential satisficing multi-core track. A comparison of the performance of the winner of this track with the winner of its single-core counterpart, the sequential satisficing track, was performed. The results clearly indicate that research in this field is not yet mature and more research is needed to exploit additional computational resources.

The arrangement of any competition series is intimately linked to the desire to show progress. As in other IPCs, a study on scalability was conducted and comparisons were made with the latest versions of the winners of IPC-2008. Overall, evidence of significant progress was found in the sequential optimal and satisficing tracks. However, progress seems to be more limited in the temporal satisficing track, for which quality improved but no evidence of enhanced coverage was observed. Again, these results should be considered only with regard to the selection of problems in the last two IPCs.

To conclude, we hope that we have contributed to highlighting the importance of the IPC series.

Acknowledgements

The deterministic part of the seventh International Planning Competition would have been impossible without the collaboration of a large number of people. Therefore we wanted to include this section in appreciation of all the assistance provided by them.

First of all, we want to express our most sincere acknowledgment to all the people who submitted a planner to the competition.

Also, to those who suggested a domain to be included in the competition, even if some were not accepted in the end: Bharatranjan Kavuluri suggested the MATCHCELLAR domain; Frédéric Maris sent us the COOKING and TEMPORAL MACHINE SHOP domains; Jörg Hoffmann and Hootan Nakhost devised the NOMYSTERY specification; Ron Petrick submitted the domain CRISP; Héctor Geffner and Nir Lipovetzky coded the VISIT-ALL domain; Amanda Coles and Andrew Coles sent us the MARKET domain; Héctor Luis Palacios prepared various conformant domains such as 1-DISPOSE, DISPOSE, GRID, LOOK-AND-GRAB, PUSH-TO and CLASSICAL; Bhaskara Marthi is behind the TIDYBOT domain; Guy Shani shall be acknowledged for sending the domains COLORBALLS DOORS, MEDICALPKS150, MEDICALPKS199, SLIDING-DOORS, UNIX and WUMPUS. At last, but not least, Tomás de la Rosa worked in the FLOORTILE domain.

We do also want to explicitly express our gratitude to Derek Long for his assistance and support with the automated validation tool VAL.

Also, to Daniel L. Kovacs for making available through the wiki site of the competition a couple of manuscripts with a formal specification of PDDL 3.1.

Besides, we do feel in debt with Óscar Pérez, Jaime Pons, Roberto Fuentes and, in general, to the Laboratory of the Computer Science Department of the University Carlos III de Madrid for their assistance in installing, configuring and maintaining the cluster.

Very importantly as well, to the IPC council for providing extensive comments and offering a lot of helpful suggestions. In particular, to Malte Helmert for inviting us to his university and sharing all the material produced at the previous IPC.

To all the members of the Planning and Learning research group (PLG) of the University Carlos III de Madrid for their encouragement and, in so many cases, for being in charge of our daily duties to allow us to work in the competition. In particular, to Daniel Borrajo for all the support.

Also, Sergio Núñez and Álvaro Torralba assisted in fruitful discussions regarding the implementation details of a number of planners. Alejandro Calderón provided a lot of insightful comments about the behavior of Linux and how it affected our experiments. Other people who contributed providing further explanations of the behavior of their planners are Nir Lipovetzky, Andrew Coles, Eric Xu, Peter Kissman, Stefan Edelkamp, Jussi Rintanen, Vincent Vidal, Yanmei Hu, Mario Baioletti, Vidal Alcázar, Frédéric Maris, Richard Valenzano, Charles Gretton and Richard Dearden.

Amanda Coles and Jussi Rintanen provided a good number of comments to some preliminary versions of this paper. Also, the reviewers of this paper provided a handful of very good comments in terms both of quantity and quality. All of them greatly enhanced not only the overall presentation but also the contents. Thanks also to the editor, Gerhard Lakemeyer for his constant guidance while writing it.

Finally, we have to acknowledge the sponsorship of Decide Soluciones, iActive, the University Carlos III de Madrid and ICAPS. The hardware platform used during the competition was funded by Spanish Science Ministry under project MICIIN TIN2008-06701-C03-03 and this work has been also partially supported by the Spanish MINECO project PlanInteraction TIN2011-27652-C03-02.

Our most sincere and heartfelt thanks to all of them.

Appendix. Supplementary material

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.artint.2015.01.004.

References

[1] A. Coles, A. Coles, A. García-Olaya, S. Jiménez, C. Linares López, S. Sanner, S. Yoon, A survey of the seventh international planning competition, AI Mag. 33 (2012) 83–88.
[2] J.A. Baier, S.A. McIlraith, Planning with preferences, AI Mag. 29 (2008) 25–36.
[3] M. Järvisalo, D. Le Berre, O. Roussel, L. Simon, The international SAT solver competitions, AI Mag. 33 (2012) 89–92.
[4] R. Howey, D. Long, M. Fox, VAL: automatic plan validation, continuous effects and mixed initiative planning using PDDL, in: The Sixteenth IEEE International Conference on Tools with Artificial Intelligence, ICTAI-04, Boca Raton (Florida), United States, 2004, pp. 294–301.
[5] Á. García-Olaya, S. Jiménez, C. Linares López, The 2011 International Planning Competition, Technical report, Universidad Carlos III de Madrid, Madrid, Spain, 2011, http://hdl.handle.net/10016/11710.
[6] E. Burns, S. Lemons, W. Ruml, R. Zhou, Best-first heuristic search for multicore machines, J. Artif. Intell. Res. 39 (2010) 689–743.
[7] R. Valenzano, N. Sturtevant, J. Schaeffer, K. Buro, Simultaneously searching with multiple settings: an alternative to parameter tuning for suboptimal single-agent search algorithms, in: Proceedings of the Twentieth International Conference on Automated Planning and Scheduling, ICAPS-10, Toronto, Canada, 2010, pp. 177–184.
[8] A. Kishimoto, A. Fukunaga, A. Botea, Evaluation of a simple, scalable, parallel best-first search strategy, Artif. Intell. 195 (2013) 222–248.
[9] D. Sulewski, S. Edelkamp, P. Kissmann, Exploiting the computational power of the graphics card: optimal state space planning on the GPU, in: Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling, ICAPS-11, Freiburg, Germany, 2011, pp. 242–249.
[10] J. Rintanen, Complexity of concurrent temporal planning, in: Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS-07, Providence (Rhode Island), United States, 2007, pp. 287–295.
[11] E. Keyder, H. Geffner, Soft goals can be compiled away, J. Artif. Intell. Res. 36 (2009) 547–556.
[12] M. Fox, D. Long, D. Magazzeni, Plan-based policies for efficient multiple battery load management, J. Artif. Intell. Res. 44 (2012) 335–382.
[13] K. Tierney, A.J. Coles, A.I. Coles, C. Kroer, A. Britt, R.M. Jensen, Automated planning for liner shipping fleet repositioning, in: Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS-12, Atibaia, São Paulo, Brazil, 2012, pp. 279–287.
[14] P. Cheeseman, B. Kanefsky, W.M. Taylor, Where the *Really* hard problems are, in: Proceedings of the Twelfth International Conference on Artificial Intelligence, IJCAI-91, Sydney, Australia, 1991, pp. 331–337.
[15] T. Bylander, A probabilistic analysis of propositional STRIPS planning, Artif. Intell. 81 (1996) 241–271.
[16] J.K. Slaney, S. Thiébaux, On the hardness of decision and optimisation problems, in: Proceedings of the Thirteenth European Conference on Artificial Intelligence, ECAI-98, Brighton, United Kingdom, 1998, pp. 244–248.
[17] J. Rintanen, Phase transitions in classical planning: an experimental study, in: Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference, KR-04, Whistler (British Columbia), Canada, 2004, pp. 710–719.
[18] J. Hoffmann, S. Edelkamp, S. Thiébaux, R. Englert, F. Liporace, S. Trüg, Engineering benchmarks for planning: the domains used in the deterministic part of IPC-4, J. Artif. Intell. Res. 26 (2006) 453–541.
[19] M. Roberts, A. Howe, Learning from planner performance, Artif. Intell. 173 (2009) 536–561.
[20] F. Calimeri, G. Ianni, T. Krennwallner, F. Ricca, The answer set programming competition, AI Mag. 33 (2012) 114–118.
[21] T. Bylander, The computational complexity of propositional STRIPS planning, Artif. Intell. 69 (1994) 165–204.
[22] M. Helmert, Complexity results for standard benchmark domains in planning, Artif. Intell. 143 (2003) 219–262.
[23] M. Helmert, New complexity results for classical planning benchmarks, in: Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling, ICAPS-06, Ambleside, The English Lake District, United Kingdom, 2006, pp. 52–61.
[24] J. Hoffmann, Where 'ignoring delete lists' works: local search topology in planning benchmarks, J. Artif. Intell. Res. 24 (2005) 685–758.
[25] J. Hoffmann, Analyzing search topology without running any search: on the connection between causal graphs and h^+, J. Artif. Intell. Res. 41 (2011) 155–229.
[26] M. Helmert, R. Mattmüller, Accuracy of admissible heuristic functions in selected planning domains, in: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI-08, Chicago (Illinois), United States, 2008, pp. 938–943.
[27] W. Cushing, S. Kambhampati Mausam, When is temporal planning really temporal?, in: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, IJCAI-07, Hyderabad, India, 2007, pp. 1852–1859.
[28] M. Helmert, G. Röger, How good is almost perfect?, in: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI-08, Chicago (Illinois), United States, 2008, pp. 944–949.
[29] S. Richter, M. Westphal, The LAMA planner: guiding cost-based anytime planning with landmarks, J. Artif. Intell. Res. 39 (2010) 127–177.
[30] A.E. Gerevini, A. Saetti, I. Serina, Planning through stochastic local search and temporal action graphs in LPG, J. Artif. Intell. Res. 20 (2003) 239–290.
[31] J. Hoffmann, B. Nebel, The FF planning system: fast plan generation through heuristic search, J. Artif. Intell. Res. 14 (2001) 253–302.
[32] M.E. Glickman, Parameter estimation in large dynamic paired comparison experiments, Appl. Stat. 48 (1999) 377–394.
[33] J. Hoffmann, S. Edelkamp, The deterministic part of IPC-4: an overview, J. Artif. Intell. Res. 24 (2005) 519–579.
[34] R.E. Bryant, Graph-based algorithms for Boolean function manipulation, IEEE Trans. Comput. 35 (1986) 677–691.
[35] M. Helmert, P. Haslum, J. Hoffmann, Flexible abstraction heuristics for optimal sequential planning, in: Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS-07, Providence (Rhode Island), United States, 2007, pp. 176–183.
[36] R. Nuzzo, Scientific method: statistical errors, Nature 506 (February 2014) 150–152.
[37] D. Long, M. Fox, The 3rd international planning competition: results and analysis, J. Artif. Intell. Res. 20 (2003) 1–59.
[38] A.E. Gerevini, P. Haslum, D. Long, A. Saetti, Y. Dimopoulos, Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners, Artif. Intell. 173 (2009) 619–668.
[39] M. Fox, D. Long, PDDL2.1: an extension to PDDL for expressing temporal planning domains, J. Artif. Intell. Res. 20 (2003) 61–124.
[40] W. Cushing, D.S. Weld, S. Kambhampati, Mausam, K. Talamadupula, Evaluating temporal planning domains, in: Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS-07, Providence (Rhode Island), United States, 2007, pp. 105–112.
[41] B.R. Kavuluri, Required concurrency without a scheduler, in: Proceedings of the Twenty-Eighth Workshop of the UK Special Interest Group on Planning and Scheduling, PlanSig-10, Brescia, Italy, 2010, pp. 71–78.
[42] D.E. Smith, D.S. Weld, Temporal planning with mutual exclusion reasoning, in: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, vol. 1, IJCAI-99, Stockholm, Sweden, 1999, pp. 326–333.
[43] Y. Hu, M. Yin, D. Cai, On the discovery and utility of precedence constraints in temporal planning, in: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, San Francisco (California), United States, 2011, pp. 1788–1789.

[44] C. Linares López, S. Jiménez, M. Helmert, Automating the evaluation of planning systems, AI Commun. 26 (2013) 331–354.
[45] J. Cohen, A power primer, Psychol. Bull. 112 (1992) 155–159.
[46] C. Bäckström, P. Jonsson, S. Ståhlberg, Fast detection of unsolvable planning instances using local consistency, in: Proceedings of the Sixth Annual Symposium on Combinatorial Search, SOCS-13, Leavenworth (Washington), United States, 2013, pp. 29–37.
[47] J. Hoffmann, P. Kissmann, A. Torralba, "Distance"? who cares? tailoring merge-and-shrink heuristics to detect unsolvability, in: Proceedings of the Twenty First European Conference on Artificial Intelligence, ECAI-14, Prague, Czech Republic, 2014, pp. 441–446.
[48] J. Hoffmann, A tough nuts track for the IPC, in: The ICAPS-07 Workshop International Planning Competition: Past, Present and Future, Providence (Rhode Island), United States, 2007.