The deterministic part of the seventh International Planning Competition – Appendices

Carlos Linares López[1,*], Sergio Jiménez Celorrio[1], Ángel García Olaya[1]

[a]*Computer Science Department, Universidad Carlos III de Madrid. Leganés (Madrid) Spain*

Abstract

This document provides additional information about the techniques used and the results obtained in the seventh International Planning Competition, IPC. As in other editions of the IPC, it contains a glossary of the different entrants that took part in the Competition and the benchmarking suite selected for their evaluation. The last three appendices cover different subjects: once the competition was over new experiments were performed to provide additional data about a couple of interesting cases; Appendix D proposes a novel technique to rank the difficulty of planning tasks that are reused from previous competitions. Finally, Appendix E provides additional explanations about the different statistical tests that were applied in the evaluation of the results produced at the IPC-2011.

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

1. Introduction

This document introduces the appendices of the analysis of the deterministic part of the seventh International Planning Competition [1]. References to different sections refer to the aforementioned paper.

The appendices are structured as follows: Appendix A succinctly describes the participants of all tracks. Appendix B provides additional details about the selected domains. Appendix C provides the results of additional experiments performed once the competition was over. Appendix D shows a novel approach used to select problems that were reused from previous IPCs and Appendix E introduces additional details about the way various statistical tests were conducted.

*Corresponding author

Email addresses: carlos.linares@uc3m.es (Carlos Linares López), sjimenez@inf.uc3m.es (Sergio Jiménez Celorrio), agolaya@inf.uc3m.es (Ángel García Olaya)

A. Planners

The entrants of the competition are summarized below. Different Sections have been arranged to differentiate among the different tracks.

A.1. Sequential optimal track

In total, twelve planners entered the sequential optimal track

BJOLP**.** (Carmel Domshlak, Malte Helmert, Erez Karpas, Emil Keyder, Silvia Richter, Gabriele Röger, Jendrik Seipp and Matthias Westphal). The BJOLP planner is built on top of the FAST DOWNWARD planning system. This planner implements the admissible heuristic h_{LA} [2] to guide a variation of the best-first search algorithm, referred as LM-A*. The h_{LA} heuristic is derived from landmarks, in particular the heuristic estimate is built using uniform cost-partitioning which ensures that the cost of each landmark is no greater than the cost of its possible achievers [3].

CPT4**.** (Vincent Vidal). CPT4 is an improved version of the optimal temporal planner created for IPCs of 2006 and 2008. Given that the optimal-temporal track of the IPC-2011 was canceled out, CPT4 entered in all tracks to evaluate how far it was from the specialized planners that participated in each track. Like its earlier versions, the CPT4 planner encodes the planning problem as a temporal constraint satisfaction problem and uses a constraint programming engine specialized to solve problems coming from the compilation of planning problems.

FAST DOWNWARD AUTOTUNE**.** (Chris Fawcett, Malte Helmert, Holger Hoos, Erez Karpas, Gabriele Röger and Jendrik Seipp). This planner applies *automated parameter tuning* to configure the FAST DOWNWARD planning system. This system has the following parameters to define: the search algorithm, the heuristic function (admissible for the sequential optimal track), and functions for combining the heuristic estimates. The configuration of FAST DOWNWARD AUTOTUNE submitted to the optimal track was found by an Iterated Local Search in the space of parameters that optimizes the mean runtime to find an optimal plan on a set of 2000 training instances from the FAST DOWNWARD benchmark set.

FAST DOWNWARD STONE SOUP**.** (Malte Helmert, Gabriele Röger, Jendrik Seipp, Erez Karpas). This entry is a sequential portfolio built on top of the search algorithms and heuristics implemented in FAST DOWNWARD. Two versions of the portfolio were submitted to the sequential optimal track. The first version included BJOLP (with 455 secs of running time), LM-CUT (569 secs), M&S1 (175 secs) and M&S2 (432 secs). The second version assigned equal portions of running time to blind search, LM-CUT, BJOLP, and the two M&S variants to avoid overfitting to the training instances. In both versions no information between the component solvers was shared and search ended as soon as one of the solvers found a solution.

FORKINIT**,** IFORKINIT**,** LMFORK**.** (Michael Katz and Carmel Domshlak). These three planners implement two implicit abstraction heuristics (*forks-only* and *inverted forks-only*) on top of the FAST DOWNWARD planning framework. Both heuristics are computed as an admissible additive combination of the true goal distances of abstract tasks. Such abstract tasks are tractable fragments of the optimal planning task identified looking at the tasks causal

graph. FORKINIT and IFORKINIT invoke the A* heuristic search algorithm while LMFork invokes LM-A*.

GAMER. (Peter Kissmann and Stefan Edelkamp). This planner is the new version of the winner of the IPC-2008 sequential optimal track that used symbolic search with BDDs. This planner improves its previous version in three ways: (1) optimizes the variable ordering in the BDDs to reduce their size, (2) implements a new method for the construction of search heuristics with the form of symbolic partial pattern databases, and (3) includes a decision procedure for the amount of bidirection in the symbolic search process.

LM-CUT. (Malte Helmert and Carmel Domshlak). This planner implements A* search guided by the landmark-cut heuristic (h^{LM-cut}) [4]. This heuristic is an admissible estimate derived by *justifying* the h_{max} values of the facts of the planning task by linking each effect of an action to its the most expensive precondition. The planner is built on top of the FAST DOWNWARD framework.

MERGE AND SHRINK. (Raz Nissim, Jörg Hoffmann and Malte Helmert). This planner implements two sequential A* searches guided by two consistent and admissible heuristic functions. The heuristics correspond to two different strategies to create estimates using solutions distances in a smaller abstract state space. The smaller abstract state space is built in an incremental fashion, starting with a set of atomic abstractions corresponding to individual variables, then iteratively merging two abstractions and shrinking them. The planner divides the given time limit between the two searches, using 4/9 of the available time for the first one (that uses the greedy bi-simulation shrinking strategy), followed if no solution is found with the second one (that uses the DFP-gop shrinking strategy) for the remaining time. This planner is implemented on top of the FAST DOWNWARD planning system.

SELMAX. (Carmel Domshlak, Malte Helmert, Erez Karpas and Shaul Markovitch). The SELMAX planner implements a LM-A* search guided by a combination of two admissible heuristics (h^{LM-cut} and h_{LA}) while still computing just a single heuristic for each search state [5]. SELMAX implements an online learning approach to deduce a decision rule that determines the heuristic to compute at each state. The learning examples are collected by sending a set of *probes* from the initial state. This planner is developed reusing the BJOLP planner, which is built on top of FAST DOWNWARD planning system.

A.2. Sequential satisficing track

Twenty seven planning systems took part in the sequential satisficing track.

ACOPLAN **and** ACOPLAN2**.** (Marco Baioletti, Alfredo Milani, Valentina Poggioni, Fabio Rossi). These two planners apply the *ant colony optimization* paradigm to solve planning problems. They implement a stochastic forward search which is biased by two factors, a relaxed planning graph heuristic and a *pheromone* model that tries to optimize the quality of the plans [6]. The two planners implement the same search strategy differing only in the function used to evaluate the quality of the plans.

ARVAND**.** (Hootan Nakhost, Richard Valenzano, Fan Xie and Martin Muller). This planner is a stochastic planner based on random walks. ARVAND generates a set of n random walks of length l, evaluate the states obtained at the end of these walks with the FF heuristic and jumps to the most promising state. ARVAND repeats this process until it finds a goal state,

or fails to improve the minimum heuristic value reached after certain jumps. After termination ARVAND initiates a new search episode by restarting from the initial state. Once a solution plan is found, ARVAND implements any-time and plan-improvement strategies to improve the quality of the solution. ARVAND is built on top of the FAST DOWNWARD planning system.

BRT. (Vidal Alczar and Manuela Veloso). BRT is a stochastic planner built on top of FAST DOWNWARD that adapts the Rapidly-exploring Random Tree (RRT) algorithm to solve planning problems. BRT tries to find a path to the goals by randomly sampling the search space and joining the closest node of the current search tree to the sampled state with a base planner. BRT exploits landmarks to bias the random sampling of the search space and uses, as a base planner, a greedy best first search with lazy evaluation and guided by the FF heuristic and preferred operators.

CBP, CBP2. (Raquel Fuentetaja). These planners modify the best first search on top of the METRICFF planner. The CBP and CBP2 planners keep two open lists during the search. One list comprises the successors from the helpful actions and the states resulting from applying relaxed plans in a look-ahead strategy. The second list comprises the set of successors generated by the non-helpful actions. The CBP and CBP2 planners do not terminate after finding a solution and try improve its quality with a Branch and Bound strategy. Both planners compute the numerical heuristics and the lookahead states using relaxed planning graphs that consider increasing levels of cost [7].

DAE_YAHSP (Johann Dréo, Pierre Savéant, Marc Schoenauer and Vincent Vidal). This planner slices a planning problem into a sequence of intermediate states that must be reached in turn before satisfying the goals and solves the intermediate sub-problems with an embedded planner, YAHSP in the case of IPC-2011 [8]. DAE_YAHSP generates and optimizes the sequence of intermediate states using an evolutionary algorithm which is driven by the critical path heuristic, h^1 [9].

FAST DOWNWARD AUTOTUNE. (Chris Fawcett, Malte Helmert, Holger Hoos, Erez Karpas, Gabriele Röger and Jendrik Seipp). Two different configurations entered at this track using non-admissible heuristics. Both versions were automatically configured with a set of 2000 training instances from the FAST DOWNWARD benchmark. FD-AUTOTUNE-SATISFICING.1, was built optimizing mean plan cost after a fixed runtime of 600 secs. FD-AUTOTUNE-SATISFICING.2, is a hybrid planner obtained by inserting, at the beginning of search, a phase configured to find satisficing plans as quickly as possible. After this phase FD-AUTOTUNE-SATISFICING.2 runs the FD-AUTOTUNE-SATISFICING configuration with the second-best training quality.

FAST DOWNWARD STONE SOUP. (Malte Helmert, Gabriele Röger, Jendrik Seipp, Erez Karpas). Two variants of the system were submitted to the sequential satisficing track. Both variants were portfolios of planners that communicate, between their component planners, the quality of the best solution found to prune solutions. The first variant used 15 of the 38 possible solvers built on FAST DOWNWARD, running them between 27 and 340 seconds. The second variant was limited to the subset of solvers corresponding to greedy best-first search with either eager or lazy evaluation performed by the heuristics h^{FF}, h^{cea}, $h^{add} and h^{CG}$. In both variants the best component was the eager greedy search guided by the h^{FF} heuristic.

FORK UNIFORM. (Michael Katz and Carmel Domshlak). First this planner runs, with a time bound of 5 minutes, a lazy greedy best-first search guided by the FF heuristic and helpful actions. After that FORK UNIFORM iteratively invokes weighted A* heuristic search guided by the admissible implicit abstraction heuristics *forks* and *inverted forks* and using helpful actions from the h^{FF} heuristic to boost the search. FORK UNIFORM sets a bound on a solution cost for the next iteration from a previously found solution. Then, lazy weighted A* is run in each next iteration, decreasing the bounds to gradually improve the plan quality.

LAMA 2008, LAMA 2011. (Silvia Richter Matthias Westphal Malte Helmert). Both versions of the LAMA planner use non-binary state variables and exploit landmarks for heuristic evaluation and for generating preferred operators. These planners first run a greedy best-first search and once a plan is found, they search for progressively better solutions by running a series of weighted A* searches with decreasing weight. The cost of the best known solution is used for pruning the search, while decreasing the weight makes the search progressively less greedy, trading speed for solution quality. Search in the LAMA planners is guided using two estimators, the FF heuristic and the landmark heuristic. To this end, the LAMA planners use separate open lists for each of the different heuristics as well as separate open lists for the preferred operators of each heuristic. When choosing which state to expand next, search alternates between the different heuristics, and expands states from preferred-operator queues with higher priority than states from other queues. LAMA-2008 is largely the same system as the one that won the sequential satisficing track in the IPC-2008. However, bug fixes have been incorporated since then. LAMA-2011 is a more efficient reimplementation in the latest version of the FAST DOWNWARD planning framework besides it runs one extra iteration of greedy best-first search ignoring costs before it starts the series of weighted A* searches.

RANDWARD, LAMAR. (Alan Olsen and Daniel Bryce). Both planners randomize the computation of the h^{FF} heuristic with the aim of reducing plateaus. The randomization is implemented by random walks in the relaxed planning space which affects to the length of the generated relaxed plan but not to the reachability of goal propositions. The planners differ in that RANDWARD does not use the landmark count heuristic, but LAMAR does. RANDWARD is built on top of FAST DOWNWARD while LAMAR is built on top of LAMA 2008.

LPRPG-P. (Amanda Coles, Andrew Coles, Maria Fox and Derek Long). LPRPG-P is a planner specially designed for dealing with metric resources [10]. This planner implements standard forward heuristic search but uses linear programming to compute a more accurate relaxed plan heuristic. In particular LPRPG-P builds a linear program model of the resources and their use, under the control of actions, to produce tighter approximations of the reachable ranges of values for variables.

MADAGASCAR, MADAGASCAR-P. (Jussi Rintanen). Both planners are based on new planning as satisfiability techniques that benefit from more compact and efficient encodings of the planning problem as a Boolean formula [11, 12]. In addition MADAGASCAR-P exploits a planning-specific heuristic for SAT solving that modifies the variable selection scheme for solving planning problems reduced to SAT.

PROBE. (Nir Lipovetzky and Hector Geffner). PROBE is a complete planner that uses greedy best first search with the standard additive heuristic (h^{add}) with just one change: when a state is selected for expansion, it first launches a probe from the state to the goal. A probe [13] is an action sequence computed greedily from a seed state for achieving a serialization of the problem subgoals that is computed dynamically along with the probe. If the probe reaches the goal, the problem is solved and the solution is returned. Otherwise, the states expanded by the probe are added to the open list. Once solution is found PROBE starts weighted A* searches reducing iteratively the weight W on the heuristic term.

ROAMER. (Qiang Lu You Xu, Ruoyun Huang and Yixin Chen). Like LAMA, ROAMER uses a greedy best-first search in a first iteration to find a plan and iteratively decreasing weighted A* searches to improve the quality of the plans found. However, ROAMER assists search with a procedure to detect and exit from plateaus. ROAMER detects plateaus of a given size by checking the number of generated nodes with the same minimum evaluation value. When a plateau is detected ROAMER runs a set of random walks to scape from it. ROAMER is also built on top of the FAST DOWNWARD planning system.

SATPLANLM-C. (Dunbo Cai, Yanmei Hu and Minghao Yin). This planner is built on top of SATPLAN and like SATPLAN, it compiles the planning problem into a Boolean formula to be solved with a SAT solver, MiniSat in the case of SATPLANLM-C. SATPLANLM-C enriches the resulting formula with extra clauses corresponding to landmarks and their orderings. These clauses serve as additional constraints to the SAT problem.

YAHSP2. (Vincent Vidal). The new version of the YAHSP planner that implement heuristic search with lookahead states. Lookahead states are generated from the application of actions from relaxed plans to speed up the search process. YAHSP2 simplifies its previous implementation and extends its expressivity to cost-based and temporal planning.

The optimal temporal planner CPT4, and the temporal satisficing planners SHARAABI and POPF2 also entered in this track with the aim of evaluating how far they were from specialized sequential satisficing planners.

A.3. Sequential multi-core track

The sequential multi-core received eight different entries.

ARVAND-HERD. (Richard Valenzano, Hootan Nakhost, Martin Muller and Jonathan Schaeffer). This participant, built on top of FAST DOWNWARD, comprises 5 components: a LAMA-2008 planner and four different configurations of the ARVAND planner. ARVAND HERD assigns one processor to the LAMA-2008 component and assigns the remaining processors to the ARVAND configurations regarding their performance in previous search episodes [14]. The four ARVAND configurations differ in the length of the random walks and in their bias for the random action selection, preferring helpful actions or deferring actions that previously led to dead-ends.

AYALSOPLAN. (Juhan Ernits, Charles Gretton and Richard Dearden). AYALSOPLAN runs multiple searches, each of which implement bitstate pruning for a distinct array size in what is otherwise a frontier search procedure. AYALSOPLAN operates in two phases. First it runs in parallel on separate processors multiple incomplete searches for plan existence. Each

incomplete search is implemented by a BFS limiting the size of its open list with bit-state pruning. If a plan is found, a second phase constructs the solution plan by repeating the successful incomplete search on a single core but this time, storing all the encountered states. The planner is built on top of LAMA.

PHSFF. (Moisés Martínez Muñoz). This planner modifies the Enforce Hill Climbing search of the FF planner to decrease its greedy behavior. In particular, the PHSFF planner parallelizes the heuristic computation process allowing the evaluation of multiple successors nodes at the same time.

YAHSP2-MT. (Vincent Vidal). This planner implement a parallel version of a best-first search algorithm [15]. YAHSP2-MT runs K threads, each expanding the next best node from a shared open list and putting them in a shared closed list. YAHSP2-MT is built on top of the YAHSP2 planner.

Multi-thread versions of ACOPLAN, MADAGASCAR, MADAGASCAR-P and ROAMER-P also entered at the sequential multi-core track but there is no specific description of the techniques used to parallelize the original planning algorithms.

A.4. Temporal satisficing track

In total, eight planners entered the temporal satisficing track, succintly described below.

DAE-YAHSP. (Johann Dréo, Pierre Savéant, Marc Schoenauer and Vincent Vidal). The temporal version of this planner attempts to find a valid sequential plan using the sequential satisficing planner DAE-YAHSP. Once the plan is found DAE-YAHSP tries to deorder the solution to reduce its make-span.

LMTD. (Yanmei Hu1, Dunbo Cai and Minghao Yin). This planner implements a greedy best-first search guided by an extension of the precedence constraints contexts heuristic (h^{pcc}) to the temporal and numeric setting. Once a plan solution is found, it searches progressively for better solutions until the search is terminated. The planner is built on the TEMPORAL FAST DOWNWARD planner that participated in the IPC-2008 and

POPF2. (Amanda Coles, Andrew Coles, Maria Fox and Derek Long). This planner is an enhanced version of the POPF planner [16], that expands partial-order plans in a forward direction by recording which steps in the plan interact with a given fact or numeric variable. POPF2 implements an anytime-search that attempts to find a plan using enforced hill-climbing (EHC) and then uses WA* to improve the quality of the found plan. The search is guided by an updated temporal relaxed planning graph heuristic.

YAHSP2. (Vincent Vidal). The temporal version of YAHSP2 obtains temporal plans deordering a valid totally ordered plan found with the sequential satisficing version of YAHSP2. A multi-threaded version, YAHSP2-MT also entered the track.

SHARAABI. (Bharat Ranjan Kavuluri). It is an extension to DRIPS, a planner born as a modification of SAPA to tackle required concurrency. SAPA is modified by adding two more types of states after the application of the *at start* effects of an action: (1) a state which signifies the time just before the end of each action and (2) a state which indicates the passage of one clock tick without any action. This causes the planner to check for actions before the producers finish and also causes the planner to generate all possibilities at each clock tick.

TLP-GP. (F. Maris and P. Régnier). This planner builds the planning graph until goals are obtained and then looks for a solution plan searching backwards in the planning graph. During this back-chaining process TLP-GP uses a *Disjunctive Temporal Problem* solver (MathSat) to verify the temporal constraints. This operation is repeated until a verified solution is obtained. In case of failure, another level is added to the planning graph and the back-chaining process is restarted. TLP-GP uses a planning graph without mutexes and constructed in an atemporal manner that ignores the duration or the start-time of actions. Conflicts between actions, including mutual exclusions, are entirely managed by MathSat during the solution extraction phase.

In addition the optimal temporal planner CPT4 entered in the temporal satisficing track with the aim of evaluating how far was from specialized sequential satisficing planners.

B. Benchmarks

This Section contains information about the benchmarks used in the IPC-2011. All planning tasks are public. Their definitions, the instances selected for the competition and generators (if any) can be downloaded from the IPC-2011 subversion system[1].

For each domain, a succint description is offered along with the rationale followed to choose problems. When reusing problems from the past competition, the Glicko score (see Appendix D) was used in a good number of cases. Also, the characteristics of the problems created in terms of objects, number of goals, etc. is discussed.

B.1. Sequential Domains

In total, 14 different domains where chosen to run the competition in the sequential optimization, satisficing and multi-core tracks[2]. They are all discussed below in alphabetical order.

B.1.1. Barman

This domain was created by Sergio Jiménez. In this domain a barman robot handles a set of drink dispensers, a set of shot-glasses and a cocktail shaker to serve a desired set of shots filled with various cocktails. Actions can fill and refill shots, pour them into the shaker, clean the shots or the shaker, do shake and serve the shots. All the actions have the same cost. The distinctive feature of this domain is that the delete effects of actions encode important knowledge: robot hands can only grasp one object at a time, and glasses need to be empty and clean to be filled. Goals are in the form `(contains shot beverage)`, where beverage can be either a single ingredient or a cocktail made mixing and shaking various ingredients. Goals involving cocktails are harder to reach than goals involving just a single ingredient.

In the problems of the competition the number of goals is the number of shots minus one, which guarantees solvability. Most of the goals are cocktails, with a maximum of 1 or 2 shots made of a single ingredient per problem. The test set of problems for this domain is generated by increasing the number of shots to serve.

Problems of the sequential satisficing tracks. The first four problems had 10 shots, 4 ingredients and 8 cocktails. They differ on the number of single-ingredient shots and on the number of shots containing the same ingredients. The number of shots and cocktails were increased up to 15 shots and 11 cocktails. The number of ingredients was always fixed to 4.

Problems of the sequential optimal track. The simplest problems in this set, problems 1 to 4, had 4 shots, 3 ingredients and 3 cocktails. The number of cocktails and shots was distributed increasingly. The last four problems had 9 shots and 7 cocktails, while preserving the number of ingredients.

[1] `svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/domains`

[2] Recall that the sequential multi-core track used exactly the same planning tasks used in the sequential satisficing track

Satisficing/Multicore							
prob.	**shots**	**ingredients**	**cocktails**	**prob.**	**shots**	**ingredients**	**cocktails**
p01-04	10	4	8	p13-16	14	4	11
p05-08	11	4	9	p17-20	15	4	11
p09-12	13	4	1				
Optimal							
prob.	**shots**	**ingredients**	**cocktails**	**prob.**	**shots**	**ingredients**	**cocktails**
p01-04	4	3	3	p13-16	8	3	6
p05-08	5	3	4	p17-20	9	3	7
p09-12	6	3	5				

Table 1: Target distribution of the test problems for the sequential tracks for the BARMAN domain.

B.1.2. Elevators

This domain was introduced in the IPC-2008 by Ioannis Refanidis, inspired by the MICONIC domain of the IPC-2000 [17]. It simulates the work of a set of elevators that need to serve a number of passengers whose initial positions and destinations are known. The building is divided in equal blocks of $M + 1$ floors, with two adjacent blocks having a common floor. There are two types of elevators, fast and slow ones. Fast elevators only stop at $M/2$ floors and have a capacity of X persons. Slow elevators only serve one block, stopping at every floor, and can hold Y persons, being usually $Y < X$. In the sequential tracks, the aim is to minimize the cost of serving all passengers. The total cost increases every time an elevator moves, while boarding and un-boarding passengers is considered to be cost-free.

From the three versions of this domain provided in the IPC-2008, only the STRIPS one is used. This version uses predicates to model both the number of floors and the number of passengers inside an elevator. Numeric fluents are used only as static predicates to account for the cost of elevators moving. Goals are always in the form `(passenger-at passenger floor)`; the number of passengers equals the number of goals. To scale the difficulty of problems in this domain, the number of floors, elevators or passengers can be increased. But also increasing the number of blocks increments the difficulty as the slow elevators have to coordinate to serve passengers.

Problems of the sequential satisficing tracks. In the satisficing track of the IPC-2011, the nine most difficult instances of the IPC-2008 were reused (according to the Glicko score). A tie in the Glicko score was observed among a number of problems in the tenth position so that problem 14 was selected because it contains more objects than the others. To complete the test set, ten new problems were generated starting from the characterization of old problem 30 (current problem 10), adding 2 fast elevators, 1 slow elevator and 1 passenger. The next 4 problems increase the number of passengers by 3. Starting from problem 16, there are 40 floors, 4 fast lifts with capacity for 6 passengers, 4 slow ones for 4 passengers, and 40 passengers, increasing the number of passengers in 5 per problem. Table 2 summarizes this information for all problems. Reused problems from the IPC-2008 are shown in boldface.

Satisficing/Multicore									
prob.	**floors**	**passengers**	**fast**	**slow**	**prob.**	**floors**	**passengers**	**fast**	**slow**
p01	17	14	2	2	p11	25	40	4	4
p02	25	21	2	3	p12	25	43	4	4
p03	25	27	2	3	p13	25	46	4	4
p04	17	26	2	2	p14	25	49	4	4
p05	17	22	2	2	p15	25	52	4	4
p06	25	30	2	3	p16	40	40	4	4
p07	25	24	2	3	p17	40	45	4	4
p08	25	33	2	3	p18	40	50	4	4
p09	25	36	2	3	p19	40	55	4	4
p10	25	39	2	3	p20	40	60	4	4
Optimal									
prob.	**floors**	**passengers**	**fast**	**slow**	**prob.**	**floors**	**passengers**	**fast**	**slow**
p01	13	3	1	2	p11	13	6	2	3
p02	13	3	1	3	p12	9	5	2	2
p03	13	3	2	2	p13	9	6	1	2
p04	9	3	1	2	p14	9	6	2	2
p05	13	4	1	2	p15	9	7	1	2
p06	9	4	2	2	p16	13	5	1	2
p07	13	5	2	3	p17	13	6	1	2
p08	9	5	1	2	p18	13	6	2	2
p09	13	3	2	3	p19	13	4	1	3
p10	13	4	2	2	p20	13	4	2	3

Table 2: Target distribution of the test problems for the sequential tracks for the ELEVATORS domain. Problems in bold are reused from the IPC-2008.

Problems of the sequential optimal track. From the IPC-2008 results, it seems that in the optimal setting the difficulty lies in the number of passengers and elevators rather than in the number of floors. Since this was a difficult domain at that time, we decided to reuse problems: problems 1 and 2 were discarded since they were too easy, and the next 20, in increasing order of their Glicko score, were selected. Table 2 summarizes the selection.

B.1.3. Floortile

This domain was created by Tomás de la Rosa. In this domain a set of robots use different colors to paint patterns in a grid of $m \times n$ floor tiles. The robots can move around the floor tiles in four directions: up, down, left and right. Robots paint with one color at a time, but can change their spray guns to any available color. However, robots can only paint the tile that is in front (up) and behind (down) them, and once a tile has been painted no robot can stand on it. For the IPC-2011 test set, robots need to paint a grid with a black and white pattern, where the cell color is alternated always, forming a kind of chess board. This particular configuration makes the domain hard because robots should paint tiles only in front of them, since painting behind leads to a dead-end. Actions allow robots to paint tiles (up and down them), to change

the color of the spray gun and to move. Each action has a different cost. It has to be noticed that moving up is more costly than moving in any other direction. Goals are in the form `(painted tile color)` and there are as many goals as tiles. The test set of problems for this domain is generated by increasing the size of the grid and the number of robots.

Problems of the sequential satisficing tracks. The first and second problems consisted of a 4×3 grid with 2 robots. The next problems increased the size of the grid and the number of robots. The most difficult problems consisted of a 7×7 grid with 3 robots.

Problems of the sequential optimal track. In this case the first and second problems consisted of a 3×3 grid with 2 robots. In the next problems, the size of the grid and the number of robots was increased. The most difficult problems consisted of a 7×6 grid with 3 robots.

Satisficing/Multicore							
prob.	**rows**	**columns**	**robots**	**prob.**	**rows**	**columns**	**robots**
p01-02	4	3	2	p11-12	5	5	2
p03-04	5	3	2	p13-14	6	5	3
p05-06	4	4	2	p15-16	6	6	3
p07-08	5	4	2	p17-18	7	6	3
p09-10	6	4	2	p19-20	7	7	3
Optimal							
prob.	**rows**	**columns**	**robots**	**prob.**	**rows**	**columns**	**robots**
p01-02	3	3	2	p11-12	6	4	2
p03-04	4	3	2	p13-14	5	5	2
p05-06	5	3	2	p15-16	6	5	3
p07-08	4	4	2	p17-18	7	5	4
p09-10	5	4	2	p19-20	7	6	3

Table 3: Target distribution of the test problems for the sequential tracks for the FLOORTILE domain.

B.1.4. Nomystery

This domain was proposed by Jörg Hoffmann and Hootan Nakhost [18, 19] to assess the performance of planners in problems with constrained resources:

> "An ubiquitous feature of planning problems is the need to economize limited resources such as fuel or money. While heuristic search, mostly based on relaxation heuristics, is currently the superior method for most varieties of planning, its ability to solve critically resource-constrained problems is limited: relaxation heuristics basically ignore the resource consumption by actions."

NOMYSTERY is a transportation domain, quite similar to the TRANSPORT domain included both in the IPC-2008 and the IPC-2011. In this domain, a truck moves in a weighted graph where a set of packages must be transported between nodes. Actions move the truck along edges and load/unload packages. Each move consumes the edge weight in fuel. In the problems used in

the competition the edge weights are uniformly drawn between 1 and an integer W. The initial fuel supply is set to $C \times M$ where M is the minimum required amount of fuel, calculated using a domain-specific optimal solver, and $C \geq 1$ is a (floating-number) input parameter that denotes the ratio between the available fuel *versus* the minimum amount required. The closer C to 1, the more constrained the problem. If $C = 1$ only the optimal plan will solve the problem. The authors provided two versions of the domain, called *Hard* and *Hard-cost* that differ only in the cost of the actions. In the first one, all the actions have unitary cost, while in the *Hard-cost* version the cost of moving a track equals the fuel consumed. According to the authors:

> "In the *Hard* encoding, problems with 12 locations and 12 packages become quite challenging for state-of-the-art planners when C is close to 1 [19]. *Hard-cost* encoding makes the problems easier for the current planners. The reason is that the heuristic functions that consider costs are not any more completely ignorant to the resource consumption of actions. However, this type of encoding is not always feasible for resource planning; there might be several resources and the cost of the plan might be different from the amount of the resource consumption. However, our initial experiments show problems in this encoding with 12 locations and 15 packages become challenging for current planners when C is close to 1."

The *Hard* version (where cost of the plan and resource consumption are not related) was chosen for the competition. As the domain creators says, this is the most challenging formulation and, besides, it does not resemble the TRANSPORT domain so much like the other. As a side effect, as the fuel levels are modeled using predicates instead of numeric fluents, this domain is also challenging for the instantiation phase of planners. As an example, the simplest problem in the sequential satisficing track has more than 3,000 predicates in its initial state. All the goals are in the form `(at package location)` where the number of packages equals the number of goals. The objective is to deliver all the packages minimizing the travel cost of the truck. There are two ways to scale problem difficulty in the NOMYSTERY domain: either increasing the number of packages (goals) and/or locations, or decreasing C.

Problems of the sequential satisficing tracks. Problems 1 through 10 start with 6 locations and 6 packages with $C = 1.5$. Each new problem adds 1 package and 1 location to the previous one, so problem 10 has 15 packages and 15 locations. Problems 11 through 20 have the same characteristics as problems 1 to 10, but in this case $C = 1.1$.

Problems of the sequential optimal track. Originally, problems 1 through 10 started with 2 locations and 1 package with $C = 1.5$ Each problem added 1 location and 1 package to the previous one, so problem 10 had 10 packages and 11 locations. Problems 11 through 20 where similar but with $C = 1.1$. This version resulted to be quite simple, so a tougher one was generated. In a second attempt, the first problem started with 4 locations and 3 packages (so that the last problem had 13 locations and 12 packages) with $C = 1.5$ and $C = 1.1$. Again, that version turned out to be quite simple, so a third one was generated. Additional experiments were performed and it was observed that the lower the constraint, the easier the planning tasks for the optimal planners. A reasonable explanation to this observation is that the search space becomes more constrained as the cost limits the number of applicable actions. Taking that into account, the new problems were generated in three different bands with $C = 1.1$ (problems 1–6), 1.5 (problems 7–13) and 2.0, for problems 14–20. Problems 1 through 6 start with 9 locations and 8 packages having as maximum 14 locations and 13 packages. Problems 7 through 13 start with 8 locations and 7

packages. The third band was exactly like the second one, the only difference being the value of C.

Satisfacing/Multicore							
prob.	**locations**	**packages**	**C**	**prob.**	**locations**	**packages**	**C**
p01	6	6	1.5	p11	6	6	1.1
p02	7	7	1.5	p12	7	7	1.1
p03	8	8	1.5	p13	8	8	1.1
p04	9	9	1.5	p14	9	9	1.1
p05	10	10	1.5	p15	10	10	1.1
p06	11	11	1.5	p16	11	11	1.1
p07	12	12	1.5	p17	12	12	1.1
p08	13	13	1.5	p18	13	13	1.1
p09	14	14	1.5	p19	14	14	1.1
p10	15	15	1.5	p20	15	15	1.1
Optimal							
prob.	**locations**	**packages**	**C**	**prob.**	**locations**	**packages**	**C**
p01	9	8	1.1	p11	12	11	1.5
p02	10	9	1.1	p12	13	12	1.5
p03	11	10	1.1	p13	14	13	1.5
p04	12	11	1.1	p14	8	7	2.0
p05	13	12	1.1	p15	9	8	2.0
p06	14	13	1.1	p16	10	9	2.0
p07	8	7	1.5	p17	11	10	2.0
p08	9	8	1.5	p18	12	11	2.0
p09	10	9	1.5	p19	13	12	2.0
p10	11	10	1.5	p20	14	13	2.0

Table 4: Target distribution of the test problems for the sequential tracks for the NOMYSTERY domain.

B.1.5. Openstacks

This domain was introduced at the IPC-2006 [20] and reused in the IPC-2008. It is based on the "minimum maximum open stacks" optimization problem:

> "A manufacturer has a number of orders, each for a combination of different products. Only one product can be made at a time, but the total required quantity of that product is made at that time. From the time that the first product requested by an order is made to the time that all products included in the order have been made, the order is said to be open and during this time it requires a stack (a temporary storage space). The problem is to order the making of the different products so that the maximum number of stacks that are in use simultaneously, i.e., the number of orders that are in simultaneous production, is minimized".

This problem is known to be NP-Hard [20], but finding non-optimal solutions is trivial. Indeed, every ordering of the sequence of made products is a solution, which will use at worst as

many stacks as orders. Finding good solutions, though, is quite hard.

Two versions of this domain were offered at the IPC-2008: STRIPS and ADL. At the IPC-2011, the STRIPS version was used. In this version, the number of stacks is represented using predicates, and universally quantified preconditions have been replaced by multiple semi-ground actions. Note that this requires a different domain file for each problem. All the goals are in the form `(shipped order)`, and the number of goals is equal to the number of orders. There are as many products as orders, but each product can be used in more than one order. Number of products making up an order is different for each order. The problem generator for this domain has a *density* parameter to establish the mean number of products in an order. The higher, the more products an order will tend to have. To scale up the difficulty, both the number of orders or their density can be increased.

Problems of the sequential satisficing tracks. The 10 most difficult problems (according to their Glicko score) were readily selected for the IPC-2011. In addition, ten new problems (from problem 11 to 20) were created. These problems came in pairs with two different values of density, 20 and 80. Problems 11 and 12 started with 130 products and this number was increased by 30 units between successive pairs.

Problems of the sequential optimal track. The first problem in the IPC-2008 started with 5 orders and each problem added an additional order, so problem 30 had 34 orders. All problems were ranked in increasing order of their Glicko score. The five easiest problems were discarded and the next 20 were selected for the IPC-2011.

Satisfacing/Multicore								
prob.	**orders**	**density**	**prob.**	**orders**	**density**	**prob.**	**orders**	**density**
p01	50	?	p08	100	?	p15	190	20
p02	60	?	p09	100	?	p16	190	80
p03	60	?	p10	100	?	p17	220	20
p04	80	?	p11	130	20	p18	220	80
p05	80	?	p12	130	80	p19	250	20
p06	80	?	p13	160	20	p20	250	80
p07	50	?	p14	160	80			
Optimal								
prob.	**orders**	**density**	**prob.**	**orders**	**density**	**prob.**	**orders**	**density**
p01	10	?	p08	17	?	p15	23	?
p02	11	?	p09	18	?	p16	25	?
p03	12	?	p10	26	?	p17	24	?
p04	13	?	p11	19	?	p18	27	?
p05	14	?	p12	20	?	p19	28	?
p06	15	?	p13	21	?	p20	29	?
p07	16	?	p14	22	?			

Table 5: Target distribution of the test problems for the sequential tracks for the OPENSTACKS domain. Problems in bold are reused from the IPC-2008.

B.1.6. Parcprinter

This domain models the operation of a multi-engine printer, developed at the Palo Alto Research Center (PARC) [21]. The compilation into PDDL was done by Rong Zhou and was first used at the IPC-2008. According to the authors, it represents the first successful industrial application of embedded domain-independent temporal planning. The real version used in practice is the temporal one, but the sequential version of the domain has been also used in the IPC.

> "This type of printer can handle multiple print jobs simultaneously. Multiple sheets, belonging to the same job or different jobs, can be printed simultaneously using multiple Image Marking Engines (IME). Each IME can either be color, which can print both color and black&white images, or mono, which can only print black&white images. Each sheet needs to go through multiple printer components such as feeder, transporter, IME, inverter, finisher and need to arrive at the finisher in order. Thus, sheet $(n+1)$ needs to be stacked in the same finisher with sheet n of the same job, but needs to arrive at the finisher right after sheet n (no other sheet stacked in between those two consecutive sheets). Given that the IMEs are heterogeneous (mixture of color and mono) and can run at different speeds, optimizing the operation of this printer for a mixture of print jobs, each of them is an arbitrary mixture of color/b&w pages that are either simplex (one-sided print) or duplex (two-sided print) is a hard problem."

A total of three printers were modeled, two of them having 2 IMEs: one color and one monochrome. The third one has 4 IMEs, two of each class. Two of the printers have a symmetric design while the other one is asymmetric. This means there are three types of domain files for this domain. Although these specific printers do not actually exist, the hardware to make them is real. In the shake of simplification all the problems simulated the printing of only a single job with multiple sheets, either one-side print or two-sides, some of them black and white and some of them color. Printer number one is not able to turn a page once printed, so if a problem prints something in the back side, the sheet needs to finish with back side up. Each action has a different cost. Printing using a color IME is more expensive than using a black and white one, but it may be worth to use the former if the feeder where the sheet lies is close to the color IME. In the initial state all sheets are blank on both sides. The goals require for each sheet to be printed either on the front, the back or both sides, and also whether the image should be printed in black and white or color. In addition, goals state which side has to be up at the end and the finishing tray where the work has to be stacked. This means that the number of goals is six times the number of sheets, though some of them are already present in the initial state. To scale up difficulty new sheets can be added to the problem or one-sided sheets can be made two-sided. There is no generator for this domain, so new problems were generated by hand modifying the IPC-2008 ones. New problems add some sheets to old problems or make two-sided some sheets.

Problems of the sequential satisficing tracks. In the IPC-2008 the first problem had 1 sheet, second one 2 sheets and so on until problem 10. The same applies for problems 11 to 20, and 21 to 30, but using different printers. All these problems were ranked in increasing order of their Glicko score and the ten most difficult instances were selected. From a closer inspection of the results of the preceding competition it was observed that printers 1 (*upp*) and 3 (*eTipp-2*) are used in problems that were more difficult than those using printer 2 (*eTipp-1*): out of the 10 most difficult problems, 5 belong to printer 3, 4 to printer 1 and only 1 to printer 2. Thus, 3 problems

were generated for printer 1, 3 for printer 2 and 4 for printer 3. New problems add sheets or sides to the former ones.

Problems of the sequential optimal track. In this track, all problems were reused from the last IPC: the four problems with lowest Glicko score were rejected and the 20 next ones (with the last four being unsolved at the IPC-2008) were selected for the IPC-2011.

Satisfacing/Multicore							
prob.	**printer**	**sheets**	**images**	**prob.**	**printer**	**sheets**	**images**
p01	upp	7	7	p11	upp	7	9
p02	upp	8	8	p12	upp	12	12
p03	upp	10	10	p13	upp	15	15
p04	eTipp-1	10	11	p14	eTipp-1	12	13
p05	eTipp-2	6	6	p15	eTipp-1	10	13
p06	eTipp-2	7	7	p16	eTipp-1	12	15
p07	upp	9	9	p17	eTipp-2	7	9
p08	eTipp-2	8	8	p18	eTipp-2	8	11
p09	eTipp-2	9	11	p19	eTipp-2	9	14
p10	eTipp-2	10	11	p20	eTipp-2	12	13
Optimal							
prob.	**printer**	**sheets**	**images**	**prob.**	**printer**	**sheets**	**images**
p01	eTipp-2	2	2	p11	eTipp-2	6	6
p02	upp	2	2	p12	eTipp-1	4	4
p03	eTipp-1	2	2	p13	upp	7	7
p04	upp	4	4	p14	upp	8	8
p05	eTipp-2	3	3	p15	upp	9	9
p06	upp	6	6	p16	upp	10	10
p07	upp	5	5	p17	eTipp-1	5	5
p08	eTipp-2	4	4	p18	eTipp-1	6	6
p09	eTipp-1	3	3	p19	eTipp-1	7	7
p10	eTipp-2	5	6	p20	eTipp-2	7	7

Table 6: Target distribution of the test problems for the sequential tracks for the PARCPRINTER domain. Problems in bold are reused from the IPC-2008.

B.1.7. Parking

This domain was first introduced by the organizers of the learning part of the IPC-2008. The domain involves parking cars on a street with N curb locations. Cars in a curb can be double-parked but not triple-parked thus ignoring the delete effects of actions implies missing relevant knowledge. The goal is to find a plan to move from one configuration of parked cars to another configuration, by driving cars from one curb location to another. The problems in the competition contain $2 \times (N - 1)$ cars, which allows one free curb space and guarantees solvability. Actions move cars from curbs or car sides to curbs or car sides and all the actions have equal cost. The number of goals equals the number of cars, and all the goals are in the form `(at-curb car`

`curb)` or `(behind-car car car)`. The test sets of problems for this domain is generated by increasing the number of cars and curb locations. Problems with the same number of cars and curbs can be quite different in difficulty as difficulty depends also on the differences between the original and goal configurations.

Problems of the sequential satisficing tracks. For this track, 6 types of problems were created, which differ in the number of cars and curbs. Problems 1 and 2 had 22 cars and 12 curbs. Next four problems had 24 cars and 13 curbs. The next 12 problems were also grouped in sets of four, each set adding 2 cars and 1 curb to the previous one. The last two problems had 32 cars and 17 curbs.

Problems of the sequential optimal track. Problems were grouped like in the satisficing track, but in this case, the first two problems had 12 cars and 7 curbs. Cars and curbs were increased by 2 and 1, respectively, in groups of 4 problems. Consequently, the last two problems had 22 cars and 12 curbs.

Satisfacing/Multicore								
prob.	**cars**	**curbs**	**prob.**	**cars**	**curbs**	**prob.**	**cars**	**curbs**
p01-02	22	12	p07-10	26	14	p15-18	30	16
p03-06	24	13	p11-14	28	15	p19-20	32	17
Optimal								
prob.	**cars**	**curbs**	**prob.**	**cars**	**curbs**	**prob.**	**cars**	**curbs**
p01-02	12	7	p07-10	16	9	p15-18	20	11
p03-06	14	8	p11-14	18	10	p19-20	22	12

Table 7: Target distribution of the test problems for the sequential tracks for the PARKING domain.

B.1.8. Pegsol

This domain, introduced originally in the IPC-2008, models the well known Peg solitaire game. It is a one-player board game that uses pegs (or balls) in a board with holes. There are a variety of versions of the game. In the most common, all holes, but the central one have pegs inserted. The goal is to empty the board leaving a single peg in the central hole, by moving pegs. A peg can move by jumping over an adjacent one (either horizontally or vertically, diagonal jumps are not allowed) into an empty hole adjacent to the jumped-over peg. The adjacent peg is then removed. There are only three actions: to start moving a peg, to keep on jumping over adjacent pegs, and to finish the move so that a different peg can be selected. The only action with positive cost is the one starting a move. Subsequent moves of the same peg are cost-free, so the planners need to search for the longest sequences of moves with the same peg. In all problems, the number of goals is always equal to 33, the number of holes in the board. Goals are in the form `(free hole)` or `(occupied hole)`. It has been empirically observed that it is the number of pegs, instead of the number of required moves, what makes it difficult for planners to find solutions in this domain. Nevertheless, this was an easy domain in the preceding competition and the number of difficult instances was rather limited. It was included in the IPC-2011 just to measure progress in comparison with the preceding edition of the IPC.

Satisfacing/Multicore									
prob.	**pegs**	**prob.**	**pegs**	**prob.**	**pegs**	**prob.**	**pegs**	**prob.**	**pegs**
p01	16	p05	16	p09	17	p13	14	p17	23
p02	16	p06	16	p10	17	p14	19	p18	25
p03	16	p07	17	p11	18	p15	15	p19	28
p04	16	p08	17	p12	19	p16	20	p20	32
Optimal									
prob.	**pegs**	**prob.**	**pegs**	**prob.**	**pegs**	**prob.**	**pegs**	**prob.**	**pegs**
p01	15	p05	16	p09	17	p13	18	p17	23
p02	16	p06	16	p10	17	p14	19	p18	25
p03	16	p07	16	p11	17	p15	19	p19	28
p04	16	p08	17	p12	17	p16	20	p20	32

Table 8: Target distribution of the test problems for the sequential tracks for the PEGSOL domain. Problems in bold are reused from the IPC-2008.

Various problems, in general easier than the standard game, can be created by either establishing different initial configurations of pegs, or having more unfilled holes.

Problems of the sequential satisficing tracks. The same set of problems for both the sequential satisficing and sequential optimal tracks were used in the IPC-2008. They were taken from a pool of 105 problems (some of them unsolvable). Most of the difficult problems, those with more than 17 pegs, were selected for the IPC-2008. In the satisficing track of the IPC-2011, 15 problems were reused and 5, taken from the same pool, were added.

Problems of the sequential optimal track. Fourteen problems were reused and 6, taken from the pool of problems, were added. The rationale was to select the most difficult problems, according to the number of pegs. Only the last problem, remains unsolved to date.

B.1.9. Scanalyzer

This domain was created by Malte Helmert, Hauke Lasinger and Robert Mattmüller for the IPC-2008 [22]. It models the problem of automatic greenhouse logistic management. Structurally it is a permutation problem with similarities to other problems like Rubik's Cube or TopSpin. At the same time, it is a real application domain. It models the LemnaTec scanalyzer platform, a system of conveyor belts designed to automatically collect data and infrared, visible and fluorescence images, of growing crops inside one or more greenhouses. This platform allows to build a *smart greenhouse* to research how to increase the growth and resistance of crops. The modeled domain consists of a series of parallel belts where batches of plants lie. The whole content of two belts can be swapped using a inner system of belts. Some of the belts, and at least the first and last ones, are also connected to the imaging chamber, where plants are analyzed. In an analysis cycle, plants start on a belt, pass through the analysis chamber and end in a different belt. Consequently, plants in the second belt pass to the first belt as well, though they are not analyzed. The domain has actions to rotate batches and to analyze them, the second action being more costly than the first one. It has also actions that rotate and analyze half batches, which makes the problem more difficult. The goal is to have all batches analyzed. For each batch there

are two goals, in the form `(analyzed batch)` and `(in-belt batch)`. To scale difficulty the number of belts can be increased. But also problems with half batches are more difficult than those considering full batches [22] and further division of belts contents in thirds, quarters, etc., not implemented in the original domain, will increase the difficulty significantly.

Problems of the sequential satisficing tracks. In the IPC-2008, the set of planning instances of the sequential satisficing and optimal tracks were the same. Problems 1 to 21 used full batches while problems 22 to 30 dealt with half batches. The first 21 problems were organized in triplets. The first problem had 6 belts, each one connected to another 3, and 6 batches. All the belts were connected to the imaging chamber. In the next problem of each triplet only half of the belts were connected to the imaging chamber, while in the last problem only the first belt entered into the imaging chamber. The next triplets were similar but they increased by 1 the number of belts. If has to be noticed that to allow half batches, each belt and batch were modeled using two objects. The domain was difficult in the IPC-2008 so instead of creating new problems, the 20 most difficult problems (according to their Glicko score) were selected. Table 9 summarizes the main characteristics of the selected problems in terms of batches, belts and total number of connections between belts. The number of connections passing through the analysis chamber is shown in parenthesis in the connections column.

Problems of the sequential optimal. Thirteen problems remained unsolved in the preceding competition. All problems were ranked in increasing order of their Glicko score and the first one was removed because it was too easy. The next 20 planning instances were then readily reused.

B.1.10. Sokoban

This domain, created in 1981 by Hiroyuki Imabayashi, was originally introduced in the IPC-2008. The decision problem of the original game was shown to be NP-hard [23, 24] but this result was refined later and it is now known to be PSPACE-complete [25].

Sokoban is a type of transport puzzle in which the player has to move a series of boxes to their final positions. Each box can only be pushed, and not pulled, in one of the four cardinal directions. As a consequence, if a box ends in a corner, it cannot be moved (so this domain has dead-ends); if it is pushed against a wall, it cannot be moved away from the wall, only displaced parallel to it.

In the IPC version of this game only the push actions have cost, moving the man around the boxes is cost-free. Goals are in the form `(at-goal box)`; the number of boxes is the number of goals. An easy way to scale up problems is to increase the number of boxes, but the problem geometry and the interaction between boxes and the boxes and the man are also relevant. Often, boxes have to be moved to temporary positions to allow other boxes to reach their final locations.

Problems of the sequential satisficing tracks. The performance of planners in the preceding competition was rather poor: the best planner solved all problems but 5, and 4 instances were not solved by any planner. At the IPC-2011, the twenty most difficult instances, according to their Glicko score, were reused. In all cases, the number of boxes ranges from 1 to 12, with a median equal to 5.

Problems of the sequential optimal track. In the IPC-2008, the last ten problems (i. e., from 21 to 30) could not be solved by any entrant. The problems of the preceding competition were ranked in increasing order of their Glicko score. The three instances with the lowest score were

Satisfacing/Multicore							
prob.	**batches**	**belts**	**connections**	**prob.**	**batches**	**belts**	**connections**
p01	8	4	8(4)	p11	14	14	50(1)
p02	12	12	72(36)	p12	12	12	42(6)
p03	8	8	17(1)	p13	10	10	30(5)
p04	10	10	26(1)	p14	8	8	72(8)
p05	14	14	98(49)	p15	16	16	65(1)
p06	8	4	6(2)	p16	18	18	90(9)
p07	16	16	128(64)	p17	18	18	82(1)
p08	8	4	5(1)	p18	12	6	18(9)
p09	12	2	37(1)	p19	12	6	12(3)
p10	14	14	56(7)	p20	12	6	10(1)
Optimal							
prob.	**batches**	**belts**	**connections**	**prob.**	**batches**	**belts**	**connections**
p01	4	2	2(1)	p11	8	4	8(4)
p02	6	6	12(3)	p12	8	8	17(1)
p03	6	6	10(1)	p13	8	4	5(1)
p04	8	8	32(16)	p14	8	4	6(2)
p05	10	10	50(25)	p15	10	10	30(5)
p06	8	8	20(4)	p16	12	12	42(6)
p07	18	18	162(81)	p17	14	14	56(7)
p08	12	12	72(36)	p18	16	16	72(8)
p09	14	14	98(49)	p19	18	18	90(9)
p10	16	16	128(64)	p20	12	6	18(9)

Table 9: Target distribution of the test problems for the sequential tracks for the SCANALYZER domain. The number of connections passing through the analysis chamber is shown in parenthesis in the connections column. Problems in bold are reused from the IPC-2008.

discarded and the next 20 were selected. Number of boxes ranges between 1 and 11, but most of the problems had 4 or less boxes.

B.1.11. Tidybot

This domain, proposed by Bhaskara Marthi, models a household cleaning task, in which one or more robots must pick up a set of objects and put them into their goal locations. This domain models a simplified version of a real task, and it is inspired by the PR2 robot developed at Willow Garage [26]. The world is structured as a grid with navigable locations and surfaces on which objects may lie. Surfaces can be isolated in the grid (*tables*), or grouped inside U-shaped enclosures (*cupboards*). The robot can reach objects on tables from any table side, but objects inside cupboards can be only accessed through the door (the U aperture). The objective is to transport the objects from their original surfaces to their destination surfaces. To do that, the robot has one gripper, which moves relative to the robot up to some maximum radius. To use the gripper the robot needs to be parked. Existing objects block the gripper, so that it may be necessary to move one object out of the way to put another one down. Robots can carry one

Satisfacing/Multicore									
prob.	**boxes**	**prob.**	**boxes**	**prob.**	**boxes**	**prob.**	**boxes**	**prob.**	**boxes**
p01	6	p05	5	p09	1	p13	8	p17	5
p02	3	p06	6	p10	4	p14	5	p18	5
p03	7	p07	5	p11	5	p15	4	p19	5
p04	5	p08	6	p12	5	p16	12 p20	12	
Optimal									
prob.	**boxes**	**prob.**	**boxes**	**prob.**	**boxes**	**prob.**	**boxes**	**prob.**	**boxes**
p01	3	p05	3	p09	3	p13	4	p17	4
p02	3	p06	4	p10	8	p14	4	p18	11
p03	3	p07	4	p11	4	p15	4	p19	4
p04	3	p08	4	p12	1	p16	4	p20	5

Table 10: Target distribution of the test problems for the sequential tracks for the SOKOBAN domain. Problems in bold are reused from the IPC-2008.

object at a time in the gripper, but may also make use of a cart, that can hold multiple objects. Regarding its difficulty:

> "In many real-world problems, the difficulty is due to the large state space and number of objects, rather than due to complex, *puzzle-like* combinatorial constraints. Humans are able to quickly find feasible solutions in such domains, because they seem to be able to decompose the problem into separate parts and make use of the geometrical structure. This domain is thus intended to exercise the ability of planners to find and exploit structure in large but mostly unconstrained problems. Optimal reasoning in such problems is challenging for humans as well, and a secondary motivation for the domain is to test the ability to do optimal reasoning in geometrically structured worlds. The presence of the carts adds another combinatorial decision: it might be worthwhile to spend some time fetching the cart to avoid later having to go back and forth with each object."

Actions move the robot in the grid, park it before using the gripper, move the gripper to grasp and un-grasp objects, and move the cart. All the actions have the same cost. Goals are expressed in the form `(object-done object)`, and the number of goals equals the number of objects. It has to be noticed though that there can be more than a final position for an object, acting as an implicit goal disjunction. A problem with x objects can have y final destinations with $y \geq x$, meaning that some objects can have more than one valid final position. The test set of problems for this domain is generated by increasing the grid size and the number of objects, surfaces and objects destinations. Geometry also counts, similar problems in terms of grid size, surfaces and number of objects can be radically different in terms of difficulty. The automatic generator makes the number of objects equal to $n \times (m-2)^2$, where n is the number of cupboards and m is the number of surfaces inside the cupboard (the size of the cupboard). Thus, there must be at least one cupboard, though the existence of tables is not mandatory. In the problems created for the IPC-2011 $m = 4$, following the recommendations from the author. Scaling the size of the grid increases the number of literals, while scaling the number of cupboards increases the number of

objects and goals. Scaling the number of tables makes navigation more constrained, since there are more obstacles, but it also increases the chances to set objects down.

Problems of the sequential satisficing tracks. We generated problems with grid size in the ranges 9×9 to 12×12, with 1 to 3 cupboards, though most of the problems have only one cupboard. Many different problems with the same size and different geometries have been created. The characteristics of the problems created are shown in Table 11. The *goals* column states the number of goal positions in which objects can be placed, though, as said, the number of goals is always equal to the number of objects.

Problems of sequential optimal track. In this case, the size of the world of the generated problems varied between 5×5 and 9×9, there is always 1 cupboard, so there are always 4 objects, and tables range between 0 and 5.

Satisfacing/Multicore									
prob.	**size**	**tables**	**cupb.**	**goals**	**prob.**	**size**	**tables**	**cupb.**	**goals**
p01	9x9	5	1	6	p11	10x10	3	1	7
p02	9x9	6	1	6	p12	10x10	2	1	7
p03	9x9	3	1	7	p13	10x10	6	1	5
p04	9x9	3	1	6	p14	10x10	8	1	7
p05	9x9	3	1	5	p15	10x10	6	1	7
p06	9x9	6	1	6	p16	10x10	3	1	5
p07	9x9	3	1	8	p17	11x11	9	1	8
p08	9x9	3	1	6	p18	11x11	7	1	6
p09	10x10	9	1	7	p19	12x12	5	3	15
p10	10x10	9	1	7	p20	12x12	7	2	11
Optimal									
prob.	**size**	**tables**	**cupb.**	**goals**	**prob.**	**size**	**tables**	**cupb.**	**goals**
p01	5x5	0	1	4	p16	9x9	5	1	6
p02-04	6x6	0	1	4	p17	9x9	4	1	6
p05-08	7x7	0	1	4	p18	9x9	3	1	7
p09-13	8x8	0	1	4	p19	9x9	4	1	5
p14	9x9	3	1	6	p20	9x9	5	1	5
p15	9x9	2	1	7					

Table 11: Target distribution of the test problems for the sequential tracks for the TIDYBOT domain.

B.1.12. Transport

This domain was created by Malte Helmert and Matthias Westphal for the IPC-2008 and it is a variation (more interesting in terms of roadmaps and cost structures) of the *logistics* domains widely used in automated planning. In this domain a series of trucks have to pick up and deliver packages in a grid of nodes representing locations within different cities. All nodes in a city are fully connected, but different cities are connected only by a node. A vehicle can transport a number of packages simultaneously, depending on its capacity. Moving costs an amount that

depends on the length of the road. However, picking up or dropping a package cost 1. Unlike the NOMYSTERY domain, there is no constraint on the fuel to use. In addition, the cost of the roads is represented using numeric variables instead of predicates, which highly reduces the number of literals in the problem. Goals are in the form `(at package location)`; the number of packages equals the number of goals. To scale problems, the number of packages, trucks and locations within a city can be increased. Number of cities has lower influence and, in the IPC-2011, problems contain no more than 3 cities.

Problems of the sequential satisficing tracks. At the IPC-2008, the first 10 problems had only 1 city, problems 11 through 20 had 2 cities, and the last 10 problems consisted of 3 cities. In the single-city problems nodes ranged from 5 to 50, while in the other two cases there were between 6 and 60 nodes. All the cities had the same number of nodes, so the number of nodes had to be multiple of 2 or 3, depending on the case. Number of trucks varied between 2 and 4, and packages between 2 and 20. Capacity of trucks was 4 in all problems.

The 10 most difficult problems, according to their Glicko score, were reused in the IPC-2011. In addition, 3 new problems were created with one city; another three with two cities and, to complete the test set, 4 new problems with three cities were added. Number of objects in these problems started from parameters of old problems 10 (one city), 20 (two cities) and 30 (three cities). The first new single-city problem increased packages, the second one extended the number of nodes and the third one increased both. The same was done for the two and three cities problems. The number of trucks was always 4.

Problems of the sequential optimal track. This domain was quite challenging for the optimal planners of the IPC-2008 and 19 out of the 30 problems remained unsolved by any entrant. After ranking all problems in increasing order of their Glicko score, the first five problems were removed because they were too easy. Only 6 of the next 20 problems were eventually solved by some planners in the preceding competition so that the difficulty was intentionally lowered by inserting new problems. However, 5 unsolved tasks from the IPC-2008 were selected also.

An analysis of the results of the previous optimal track showed that the most important parameters to set the difficulty of the problems seemed to be the number of nodes and packages (the effect of trucks could not be measured as no problem with 3 trucks was solved). It seemed that planning tasks with a number of nodes ranging between 12 and 16 started to be challenging. Thus, 10 new problems were generated: six of them were one-city problems with 13, 14 and 15 nodes, 2 or 3 trucks and from 6 to 8 packages; 2 of them were two-cities problems with 14 nodes, between 2 and 3 trucks and 8 packages; the last two tasks were three-cities problems with 15 nodes, 2 or 3 trucks and 9 packages.

Satisfacing/Multicore									
prob.	**cities**	**nodes**	**trucks**	**pkg.**	**prob.**	**cities**	**nodes**	**trucks**	**pkg.**
p01	1	40	4	16	p11	1	50	4	22
p02	1	45	4	18	p12	1	53	4	20
p03	3	54	4	18	p13	1	53	4	22
p04	2	36	4	12	p14	2	60	4	22
p05	2	42	4	14	p15	2	62	4	20
p06	2	48	4	16	p16	2	62	4	22
p07	2	54	4	18	p17	3	60	4	22
p08	1	50	4	20	p18	3	63	4	20
p09	2	60	4	20	p19	3	63	4	22
p10	3	60	4	20	p20	3	66	4	22
Optimal									
prob.	**cities**	**nodes**	**trucks**	**pkg.**	**prob.**	**cities**	**nodes**	**trucks**	**pkg.**
p01	3	9	2	4	p11	2	14	2	8
p02	1	9	2	4	p12	2	14	3	8
p03	2	8	2	3	p13	1	15	2	8
p04	2	12	2	4	p14	1	15	3	8
p05	3	12	2	5	p15	3	15	2	9
p06	1	12	2	5	p16	3	15	3	9
p07	1	13	2	6	p17	1	15	2	6
p08	1	13	3	6	p18	2	16	2	5
p09	1	14	2	7	p19	3	15	2	6
p10	1	14	3	7	p20	1	18	3	7

Table 12: Target distribution of the test problems for the sequential tracks for the TRANSPORT domain. Problems in bold are reused from the IPC-2008.

B.1.13. VisitAll

This domain was created by Nir Lipovetzky and Héctor Geffner, to evaluate the performance of planners in problems with *conflicting* goals. It is somehow inspired by the DISPOSE domain created by Héctor Palacios and Héctor Geffner for the conformant track of the uncertainty part of the IPC-2008 [27]. In this kind of problems progress toward one goal means moving away from other goals. Such problems typically produce large plateaux when using delete-relaxation heuristics. In the VISITALL domain, a robot in the middle of a $n \times n$ square grid must visit a number of cells in the grid. There is only one action, which moves the robot from one cell to an adjacent one. The goals are in the form `(visited cell)` and their number equals the number of cells, except for the sequential optimal track where there are problems where only half of the cells have to be visited. In addition to be challenging to the delete-relaxation heuristics, this domain also challenges the grounding capacity of planners. The only way to scale in this domain is to increase the size of the grid.

Problems of the sequential satisficing tracks. The first problem consisted of a 12×12 grid and it was increased in steps of 2 until the last problem, which resulted consequently in a 50×50 grid. In this track, all goals have to be visited.

Problems of the sequential optimal track. For the sequential optimal track, the size of the grid ranges between 2×2 and 11×11. For each size two problems were created: one, where all cells have to be visited; and another easier one, where the robot needs only to visit half of the cells.

Satisfacing/Multicore							
prob.	**size**	**prob.**	**size**	**prob.**	**size**	**prob.**	**size**
p01	12x12	p06	22x22	p11	30x30	p16	40x40
p02	14x14	p07	24x24	p12	32x32	p17	42x42
p03	16x16	p08	26x26	p13	34x34	p18	44x44
p04	18x18	p09	28x28	p14	36x36	p19	46x46
p05	20x20	p10	30x30	p15	38x38	p20	50x50
Optimal							
prob.	**size**	**prob.**	**size**	**prob.**	**size**	**prob.**	**size**
p01-02	2x2	p07-08	5x5	p13-14	8x8	p19-20	11x11
p03-04	3x3	p09-10	6x6	p15-16	9x9		
p05-06	4x4	p11-12	7x7	p17-18	10x10		

Table 13: Target distribution of the test problems for the sequential tracks for the VISITALL domain.

B.1.14. Woodworking

This domain, created by Malte Helmert and Gabrielle Röger, was first used in the IPC-2008. It is a variant of the SCHEDULE domain of the IPC-2000 [17], focusing on action costs and interaction between objects. It simulates the works in a woodworking workshop where there are some boards of wood that have to be sawed, polished, colored, etc. to build some parts. Some of the actions have fixed costs, while the cost of others depend on the piece of wood being treated, so they depend on the problem, but not on the state. The objective is to produce the parts. For

each part to be done there is a goal in the form `(available part)`, but also goals stating the color, the polishing, the type of wood, etc. Although there is not a fixed number, usually there are between 3 and 5 goals per part. The higher the number of parts and initial boards, the more difficult the problem. However, the effect of the number of tools in the difficulty is less predictable. If a 3-parts problem with 1 tool of each type is taken, and a similar one with 3 tools of each type is generated, the solving time ranges from a few seconds to some minutes, when using delete-relaxation heuristics. But if the number of tools is increased to 5, time is reduced and becomes similar to the 1-tool problem. Increasing tools to 7 reduces the time even more.

Problems of the sequential satisficing tracks. Parameters of each problem are the parts to be done and the initial quantity of available wood (boards), expressed as a percentage of the necessary wood. In the IPC-2008, problems 1 to 10 had a 140% of wood and from 3 to 30 parts. Problems 11 to 20 and 21 to 30 were similar but with 120% and 100% of wood, respectively. All problems were ranked in increasing order of their Glicko score and the ten most difficult instances were selected. Interestingly, the Glicko score revealed that problems increased in difficulty with the available wood. Given so, new problems were generated with 120% and 140% of wood, parts ranging from 33 to 39, one tool of each type and from 20 to 23 with three tools.

Problems of the sequential optimal track. At the optimal track of the IPC-2008 the ten first problems had a 140% of wood and parts were increased in steps of one unit from one problem to another, starting from 3 parts. Similarly, problems 11–20 had 120% of wood, while problems 21–30 decreased the available wood to 100%. This setting resulted to be difficult and most planners were not able to handle problems with more than 5 parts. Only one planner was able to solve those problems, but it reported both non-optimal solutions and invalid plans for the easiest instances, so it is very likely that the found plans were also non-optimal. After ranking the problems in increasing order of their Glicko score, the first five ones were discarded because they seemed too easy. The next 20 problems in ascending order of their Glicko score were selected.

B.2. Temporal Domains

Next we describe the selection of 12 domains for the temporal satisficing track. For the domains that already appeared in the sequential tracks we only describe the specific temporal features. For further description of these domains, reader should refer to the previous section.

B.2.1. Crewplanning

This domain, introduced in the IPC-2008, is a PDDL translation of an original domain written in ANML [28] that models the activities of human spaceflight crews for the International Space Station, ISS [29]. The aim of the domain is to automatically create the *On-Board Short Term Plan* with the daily activities of the crew of the ISS during the period they remain on space. Actions include sleeping, having meal, exercising, doing payload activities and performing various maintenance activities. Each action has a different duration. Some activities have to be done daily, (sleeping, exercising...), some have fixed times, and others have to be performed only once with no specific schedule. Actions can be interleaved but there is no required concurrency. There is one goal for each activity the crew has to perform and the solution plans must accomplish all the activities in the shortest timespan. Problems scale mainly by increasing the crew members and the payload activities.

Satisfacing/Multicore							
prob.	**wood**	**parts**	**tools**	**prob.**	**wood**	**parts**	**tools**
p01	140%	21	1	p11	120%	33	1
p02	120%	24	1	p12	140%	33	1
p03	120%	27	1	p13	120%	36	1
p04	140%	27	1	p14	140%	36	1
p05	120%	30	1	p15	120%	39	1
p06	140%	30	1	p16	140%	39	1
p07	100%	24	1	p17	120%	20	3
p08	100%	27	1	p18	140%	20	3
p09	120%	24	1	p19	120%	23	3
p10	140%	21	1	p20	140%	23	3
Optimal							
prob.	**wood**	**parts**	**tools**	**prob.**	**wood**	**parts**	**tools**
p01	100%	5	1	p11	100%	8	1
p02	120%	4	1	p12	120%	8	1
p03	120%	5	1	p13	120%	8	1
p04	120%	5	1	p14	100%	9	1
p05	100%	6	1	p15	120%	9	1
p06	120%	6	1	p16	120%	9	1
p07	120%	6	1	p17	100%	10	1
p08	100%	7	1	p18	120%	10	1
p09	120%	7	1	p19	120%	10	1
p10	120%	7	1	p20	100%	11	1

Table 14: Target distribution of the test problems for the sequential tracks for the WOODWORKING domain. Problems in bold are reused from the IPC-2008.

Problems of the temporal satisficing track. Most of the planners of the IPC-2008 were not able to solve more than half of the problems of this domain. Given that, the 20 most difficult problems according to their Glicko score were selected so that no new problems were generated.

B.2.2. Elevators

This domain is a temporal version of the domain described for the sequential track (see Section B.1.2 in page 10) that transforms the cost of moving elevators into durations. At the temporal track of the IPC-2008, there were two versions of this domain, *numeric* and STRIPS. Only 2 planners participated in the *numeric* version, and since most planners of the IPC-2011 do not support numeric domains, only the STRIPS version was reused.

Problems of the temporal satisficing track. The IPC-2008 used the same set of problems, exchanging cost and durations, for the sequential satisficing and the temporal tracks. The problems were ranked in increasing order of their Glicko score and 10 most difficult instances were reused. Additionally, ten new problems were created to complete the test set. These new problems were similar in terms of elevators, floors and passengers to the 10 last problems of the sequential satisficing track —see Table 2.

prob.	crew	days	activities	prob.	crew	days	activities
p01	1	2	7	p11	2	3	36
p02	1	2	10	p12	2	3	41
p03	1	2	13	p13	2	3	38
p04	3	1	16	p14	3	3	42
p05	2	1	11	p15	3	3	67
p06	1	3	17	p16	3	3	69
p07	1	3	7	p17	2	2	21
p08	1	3	16	p18	2	2	25
p09	2	2	21	p19	3	2	35
p10	3	2	33	p20	3	2	46

Table 15: Target distribution of the test problems for the CREWPLANNING domain. Problems in bold are reused from the IPC-2008.

prob.	floors	passengers	fast	slow	prob.	floors	passengers	fast	slow
p01	17	26	2	2	p11	25	40	4	4
p02	25	15	2	3	p12	25	43	4	4
p03	25	18	2	3	p13	25	46	4	4
p04	25	21	2	2	p14	25	49	4	4
p05	25	24	2	2	p15	25	52	4	4
p06	25	27	2	3	p16	40	40	4	4
p07	25	30	2	3	p17	40	45	4	4
p08	25	33	2	3	p18	40	50	4	4
p09	25	36	2	3	p19	40	55	4	4
p10	25	39	2	3	p20	40	60	4	4

Table 16: Target distribution of the test problems for the temporal track for the ELEVATORS domain. Problems in bold are reused from the IPC-2008.

B.2.3. Floortile

This is the temporal version of the domain introduced in the sequential track, see Section B.1.3 in page 11. Actions in the temporal version are converted into durative actions making their duration equal to their cost in the sequential track. If there are more than one robot, which is the case for all the problems generated for this domain, the plan can be parallelized reducing the makespan of the solutions.

Problems of the temporal satisficing track. There were four groups of five problems with the same configuration each. Problems 1 through 5 consisted of a 4×3 grid and 2 robots. Next five problems increased by 1 the size of the grid. Problems 11 to 15 increased the number of robots, while last five problems increased again the size, having three robots on a 6×5 grid.

prob.	grid	robots	prob.	grid	robots
p01-05	4x3	2	p11-15	5x4	3
p06-10	5x4	2	P16-20	6x5	3

Table 17: Target distribution of the test problems for the temporal track for the FLOORTILE domain.

B.2.4. Matchcellar

This domain requires concurrency and describes a world with fuses to mend and matches to help in the repair. The domain has only two actions, light a match and mend a fuse. Goals are in the form `(mended fuse)` so the number of goals is the number of fuses. The domain was submitted by Bharat Ranjan Kavuluri, inspired by the MATCH domain [30]. The submitted version used fluents so it was adapted to the STRIPS version. This version requires a hand free and light from matches to mend a given fuse. Therefore, planners are forced to have light from matches while a given fuse is being mended and only planners that deal with required concurrency can solve problems in this domain. The test set of problems for this domain is generated by increasing the number of damaged fuses and available matches.

Problems of the temporal satisficing track. First problem had 6 fuses and 3 matches. Next problems increased both parameters in steps of one unit, so that the last problem consists of 25 fuses and 22 matches.

prob.	matches	fuses	prob.	matches	fuses	prob.	matches	fuses
p01	3	6	p08	10	13	p15	17	20
p02	4	7	p09	11	14	p16	18	21
p03	5	8	p10	12	15	p17	19	22
p04	6	9	p11	13	16	p18	20	23
p05	7	10	p12	14	17	p19	21	24
p06	8	11	p13	15	18	p20	22	25
p07	9	12	p14	16	19			

Table 18: Target distribution of the test problems for the temporal track for the MATCHCELLAR domain.

B.2.5. Openstacks

This is the temporal version of the domain used in the sequential track, see Section B.1.5 in page 14. In the IPC-2008 there were 4 versions for this domain but only the STRIPS version was used in the IPC-2011. In the temporal version each action has a different duration. Actions can be parallelized to reduce the makespan of the solutions but concurrency is not required.

Problems of the temporal satisficing track. At the IPC-2008 three planners, including the baseline, solved all the planning tasks. All problems were sorted in increasing order of their Glicko score and the 10 problems with higher score were selected for the IPC-2011. The set of 20 problems was completed generating 10 new problems expected to be more difficult. Starting from problem 30 of the IPC-2008 (current problem 10) orders were increased in one unit, until 44 orders were reached in problem 20. The resulting new problems had a 80% density.

prob.	orders	density	prob.	orders	density	prob.	orders	density
p01	22	?	p08	29	?	p15	39	80
p02	23	?	p09	30	?	p16	40	80
p03	24	?	p10	31	?	p17	41	80
p04	25	?	p11	35	80	p18	42	80
p05	27	?	p12	36	80	p19	43	80
p06	28	?	p13	37	80	p20	44	80
p07	28	?	p14	38	80			

Table 19: Target distribution of the test problems for the temporal track for the OPENSTACKS domain. Problems in bold are reused from the IPC-2008.

B.2.6. Parcprinter

This is the temporal version of the domain with the same name of the sequential track, see Section B.1.6 in page 16. According to the authors, the temporal track is the most natural fit for this domain. This is due to the default objective function for the printer of maximizing its productivity, which equals to finish printing all job requests as quickly as possible. In this domain, concurrency is not required to solve problems.

Problems of the temporal satisficing track. Like in the sequential tracks, new problems were generated by-hand extending the IPC-2008 test set. Accordingly, the 10 most difficult problems from the IPC-2008 were selected, according to their Glicko scores, and 10 new problems, more difficult than those, were generated to complete the test set. New problems were created by adding sheets or images to the former ones. Four problems used the *upp* printer, while three used the *eTipp-1* printer and three the *eTipp-2* one.

prob.	printer	sheets	images	prob.	printer	sheets	images
p01	upp	8	8	p11	upp	11	11
p02	upp	9	9	p12	upp	12	12
p03	upp	10	10	p13	upp	13	13
p04	eTipp-1	6	6	p14	upp	14	14
p05	eTipp-1	7	7	p15	eTipp-1	10	12
p06	eTipp-1	8	8	p16	eTipp-1	11	12
p07	eTipp-1	10	11	p17	eTipp-1	11	13
p08	eTipp-1	5	6	p18	eTipp-2	10	12
p09	eTipp-2	10	11	p19	eTipp-2	11	12
p10	eTipp-1	9	11	p20	eTipp-2	11	13

Table 20: Target distribution of the test problems for the temporal track for the PARCPRINTER domain. Problems in bold are reused from the IPC-2008.

B.2.7. Parking

This is the temporal version of the same domain of the sequential track (see Section B.1.7 in page 17) and it is used in a temporal track for the first time. In the temporal version, actions have

different durations but concurrency is not required. Plans can be parallelized by moving various cars simultaneously to different curbs which reduces the makespan of the solution plans.

Problems of the temporal satisficing track. Sets of three problems with the same number of objects but different configurations were created. The simplest triplet had 11 cars and 7 curbs. The most difficult one, actually a pair instead of a triplet, had 22 cars and 13 curbs.

prob.	**cars**	**curbs**	**prob.**	**cars**	**curbs**	**prob.**	**cars**	**curbs**
p01-03	11	7	p10-12	16	10	p16-18	20	12
p04-06	13	8	p13-15	18	11	p19-20	22	13
p07-09	15	9						

Table 21: Target distribution of the test problems for the temporal track for the PARKING domain.

B.2.8. Pegsol

This is the temporal version of the domain with the same name used in the sequential tracks, see Section B.1.8 in page 18. In the temporal version of this domain there is no distinction in time between starting a jump and continuing it, so that minimizing the makespan does not necessarily results in short sequences of continued jumps. Instead, this fact fosters parallelization, allowing to move more than one peg simultaneously. The temporal version of this domain does not require concurrency.

Problems of the temporal satisficing track. All the problems were reused from the last IPC. All planning tasks were sorted in increasing order of their Glicko score and the 20 ones with the highest score were selected.

B.2.9. Sokoban

This is the temporal version of the same domain from the sequential track, see Section B.1.10 in page 20. The main difference is that in the temporal version there are multiple men that can act in parallel, so that plans may not be fully sequential. In this temporal version concurrency is not needed but an interesting extension of this domain could be built by requiring more than one man to push a box.

prob.	**pegs**	**prob.**	**pegs**	**prob.**	**pegs**	**prob.**	**pegs**
p01	17	p06	12	p11	14	p16	25
p02	19	p07	12	p12	16	p17	18
p03	9	p08	19	p13	14	p18	16
p04	10	p09	13	p14	17	p19	16
p05	11	p10	15	p15	15	p20	32

Table 22: Target distribution of the test problems for the temporal track for the PEGSOL domain. Problems in bold are reused from the IPC-2008.

Problems of the temporal satisficing track. The performance of planners in this domain was quite poor in the IPC-2008, with 12 unsolved problems. Given that, instead of generating new problems the most difficult instances, according to their Glicko score, were reused.

prob.	**men**	**boxes**	**prob.**	**men**	**boxes**	**prob.**	**men**	**boxes**
p01	3	8	p08	17	8	p15	5	15
p02	2	4	p09	20	12	p16	13	12
p03	3	3	p10	14	13	p17	3	9
p04	3	5	p11	3	3	p18	10	6
p05	2	7	p12	5	13	p19	8	15
p06	2	4	p13	2	13	p20	2	8
p07	9	10	p14	4	9			

Table 23: Target distribution of the test problems for the temporal track for the SOKOBAN domain. Problems in bold are reused from the IPC-2008.

B.2.10. Storage

This domain was introduced in the IPC-2006 [20]. It deals with moving a certain number of crates from some containers to some depots using hoists. Each depot is divided into different areas. Inside a depot, each hoist can move according to a specified spatial map connecting different areas of the depot. Crates block hoists movements inside the depots but, outside them hoists movements are unrestricted. The test problems for this domain involve different numbers of depots, hoists, crates, containers, and depot areas. The domain has actions for lifting and dropping a crate by a hoist, and for moving a hoist into a depot, from one area of a depot to another one, and outside a depot. Actions can be parallelized when more than one hoist is available but concurrency is not required. Goals are in the form `(in crate depot)` and its number equals the number of crates. To scale problems in this domain, the number of depots, areas, containers, crates or hoists can be increased.

Problems of the temporal satisficing track. Problems were grouped into sets of five describing different configurations for the same number of objects. The first five problems consist of 1 hoist, 1 depot with 8 areas, 2 containers and 8 crates. The most difficult problems had 3 hoists, 4 depots with 4 areas each, 4 containers, and 16 crates.

prob.	**hoists**	**depots**	**containers**	**crates**	**areas**
p01-05	1	1	2	8	8
p06-10	2	2	2	8	8
p11-15	3	3	3	12	12
p16-20	3	4	4	16	16

Table 24: Target distribution of the test problems for the temporal track for the STORAGE domain.

B.2.11. TMS

This domain requires concurrency and it is inspired by a real-world application. TMS stands for Temporal Machine Shop. It was submitted by Fréderic Maris and Pierre Régnier and it is a *light* version of another domain [31]. It models the use of k kilns, each having different baking times, to bake and treat p ceramic pieces of t different types, each type requiring different baking and treatment times, being always baking time bigger than treatment time. Ceramics can be assembled to produce biggest pieces (structures), which can be baked again. The IPC-2011 version can be considered a *light* version of the original domain because it is tailored to temporal planners that do not support durative actions with time intervals. Actions have different durations and baking and treatment have to be performed concurrently (actually treatment must be done while baking) . Goals are in the form `(baked-structure piece1 piece2)`, where both pieces usually belong to different types of ceramic. The number of goals is half the number of pieces. Problems scale by increasing the number of pieces.

Problems of the temporal satisficing track. First problem had a total of 50 pieces, 10 of type A, 15 of type B and 25 of type C. This structure is maintained for the next problems but each problem increases type A pieces by 2, type B pieces by 3 and type C pieces by 5.

prob.	**type A**	**type B**	**type C**	**prob.**	**type A**	**type B**	**type C**
p01	10	15	25	p11	30	45	75
p02	12	18	30	p12	32	48	80
p03	14	21	35	p13	34	51	85
p04	16	24	40	p14	36	54	90
p05	18	27	45	p15	38	57	95
p06	20	30	50	p16	40	60	100
p07	22	33	55	p17	42	63	105
p08	24	36	60	p18	44	66	110
p09	26	39	65	p19	46	69	115
p10	28	42	70	p20	48	72	120

Table 25: Target distribution of the test problems for the temporal track for the TMS domain.

B.2.12. TurnAndOpen

This domain was created by Sergio Jiménez. In this domain there are a set of rooms containing balls and a number of robots with two gripper hands to transport them. The goal is to find a plan for the robots to transport balls from certain rooms to another ones. There are doors that must be open to move from one room to another. To open a door, a robot must turn the doorknob and push the door at the same time. Because of this fact, problems in this domain cannot be solved by sequential planners and only planners able to produce plans with concurrent actions can generate solutions. In addition, different robots can act in parallel reducing the makespan of the solutions. The test set of problems for this domain is generated by increasing the number of rooms, the number of balls to transport and the number of robots.

Problems of the temporal satisficing track. The first problem had 2 robots, 8 rooms and 10 balls to be transported. Each problem increased balls by two. Rooms were increased by one every

four problems. Meanwhile, robots increased by 1 each eight problems. Consequently the last problem had 4 robots, 12 rooms and 48 balls to transport.

prob.	**robots**	**rooms**	**balls**	**prob.**	**robots**	**rooms**	**balls**
p01	2	8	10	p11	3	10	30
p02	2	8	12	p12	3	10	32
p03	2	8	14	p13	3	11	34
p04	2	8	16	p14	3	11	36
p05	2	9	18	p15	3	11	38
p06	2	9	20	p16	3	11	40
p07	2	9	22	p17	4	12	42
p08	2	9	24	p18	4	12	44
p09	3	10	26	p19	4	12	46
p10	3	10	28	p20	4	12	48

Table 26: Target distribution of the test problems for the temporal track for the TURNANDOPEN domain.

C. Additional experiments

According to the rules of the seventh International Planning Competition (see Section 2.1), it was feasible to patch a planner provided that the fix was not relative to the core elements of the planner such as the heuristic search algorithm, the heuristic functions used to guide the search or any other sensitive part of the code. This Appendix reports the performance of two fixed versions of planners of the IPC-2011. Since these versions effectively patch a number of bugs and the final versions are readily available, the results of these versions are succinctly described here for the sake of completeness. In one particular case, the new version of the planner was submitted for evaluation after the competition was concluded. In the second case, the patch was rejected because the authors explicitly warned the organizers that they were modifying the search algorithm, where a bug was found once the competition started.

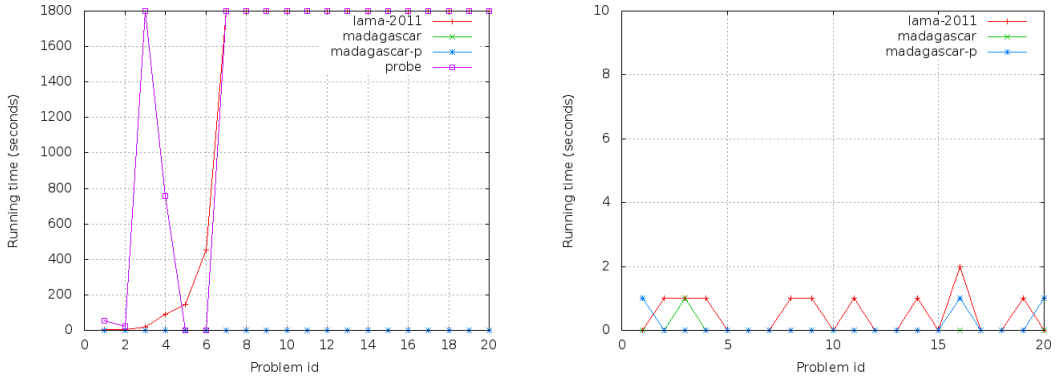
C.1. MADAGASCAR *and* MADAGASCAR-P

MADAGASCAR and MADAGASCAR-P entered the sequential satisficing and sequential multi-core tracks —see Sections 4.3 and 4.4, respectively. During the competition it was observed that these planners produced invalid solutions in the domains ELEVATORS and WOODWORKING. The author found that in some cases actions were instantiated incorrectly, ground actions could be missing and there could be even incorrect additional ones. Furthermore the author found a bug in the parser when dealing with metric declarations, terminating the planner immediately without delivering a plan. These two bugs were fixed once the competition was over. However, the difference in performance is significant and therefore the new results are reported here for the sake of completeness.

Table 27 shows the differences in coverage per domain in the sequential satisficing track. As it can be seen, both versions improve their coverage by more than 50%. Special attention deserve the domains FLOORTILE and WOODWORKING where fixing these bugs allow both planners to solve all instances when either none or only one were solved in the official competition. Additionally, MADAGASCAR-P solves almost all problems in the ELEVATORS domain.

An interesting observation is that, in spite of our efforts to create challenging problems (see Section 3.2), some domains were impressively simple for both planners once the bugs were fixed. In particular, both MADAGASCAR and MADAGASCAR-P solve all planning tasks in the FLOORTILE, PARCPRINTER and WOODWORKING domains in one second or less. Also, MADAGASCAR-P solved all problems from the domain PEGSOL in less than four seconds on average. Figure 1 shows the overall running time of MADAGASCAR and MADAGASCAR-P for solving each instance in these domains. Their performance is compared with LAMA-2011 (which is the fastest planner according to the Wilcoxon signed-rank test performed in Section 4.3.2, see Figure 11) and also with PROBE —who scores second in the *Time* metric discussed in Section 6, see Table 9. While MADAGASCAR and MADAGASCAR-P solved all planning tasks in these domains, LAMA-2011 and PROBE solve only 6 and 5 instances respectively in FLOORTILE. The overall running time for these problems are shown in Figure 1 with $t = 1800$. Additionally, PROBE failed to solve 6 cases in PARCPRINTER. While it solved the remaining cases very fast, it was still slower than either LAMA-2011 or MADAGASCAR/MADAGASCAR-P so that it has been removed to improve readability in Figure 1(b).

The difference in performance is very remarkable in FLOORTILE (Figure 1(a)) and WOODWORKING (Figure 1(d)), specially in the first one where MADAGASCAR and MADAGASCAR-P even manage to solve problems in less than 2 seconds that are unaffordable both for LAMA-2011 and PROBE in 1800 seconds. These figures endorse the observation anticipated in Sec-



(a) FLOORTILE (b) PARCPRINTER

(c) PEGSOL (d) WOODWORKING

Figure 1: Raw speed (in seconds) of LAMA-2011, PROBE, MADAGASCAR and MADAGASCAR-P to generate the first solution in a selected subset of domains

Domain	M	MP
BARMAN	0 / 0	0 / 0
ELEVATORS	1 / 0	19 / 0
FLOORTILE	20 / 0	20 / 0
NOMYSTERY	15 /17	15 /15
OPENSTACKS	0 / 0	0 / 0
PARCPRINTER	20 /20	20 /20
PARKING	0 / 0	0 / 0
PEGSOL	17 /17	20 /20
SCANALYZER	12 /11	18 /18
SOKOBAN	0 / 0	2 / 2
TIDYBOT	0 / 1	12 /10
TRANSPORT	0 / 0	2 / 2
VISITALL	0 / 0	0 / 0
WOODWORKING	20 / 1	20 / 1
Total	105/67	148/88

Table 27: Coverage of the new versions of MADAGASCAR and MADAGASCAR-P in the sequential satisficing track denoted as M and MP respectively which are also conventional names for the same planners. The coverage of the official version of the IPC is shown in second place

tion 3.2, that the selection of problems was biased towards the performance of a particular class of planners so that the difficulty for another type of planners (such as MADAGASCAR and MADAGASCAR-P) does not necessarily follow the same trends. In the other two domains, PARCPRINTER (Figure 1(b)) and PEGSOL (Figure 1(c)), MADAGASCAR and MADAGASCAR-P are clearly competitive with the best performer in the track.

C.2. ARVANDHERD

ARVANDHERD entered the sequential multi-core track (see Section 4.4) and actually won the competition in that category. However, once the competition was started the authors found a bug in their code and warned the organizers that it was relative to the implementation of the search algorithm. In short, one of the cores was specifically designed to perform several iterations of weighted A* with a decreasing weight. However, a greedy best-first search was invoked in every iteration instead. While it was not expected that fixing this bug would improve coverage too much, it could affect the quality of the final solutions. Thus, the new version was ran in *non-competitive* mode (i. e., without considering its results to be part of the official results of the competition but to be reported here for the sake of completeness) and both coverage and its score in terms of quality (see Section 2.3) are examined.

Table 28 shows the coverage per domain of both the official version used in the competition of ARVANDHERD and the same planner after fixing the bugs, ARVANDHERD-2. As anticipated by the authors, the new version only improves coverage marginally in 6 problems and, indeed, it decrements it in one problem in BARMAN and TRANSPORT.

On the other hand, coverage was found to be strongly correlated with the official metric of the competition (see Section 6). Consequently, only marginal improvements in the final score should be expected if the improved version would have officially entered the competition. To examine

Domain	ARVANDHERD	ARVANDHERD-2
BARMAN	17	16
ELEVATORS	20	20
FLOORTILE	2	6
NOMYSTERY	19	20
OPENSTACKS	20	20
PARCPRINTER	20	20
PARKING	19	19
PEGSOL	20	20
SCANALYZER	20	20
SOKOBAN	15	16
TIDYBOT	18	20
TRANSPORT	16	15
VISITALL	10	10
WOODWORKING	20	20
Total	236	242

Table 28: Coverage of the official version of ARVANDHERD and the patched version denoted as ARVANDHERD-2

this in more detail, a separate competition was run replacing ARVANDHERD by ARVANDHERD-2. Indeed, the performance of the new version is much alike its predecessor and it improves the score from 227.07 to 233.45, roughly six points, in correspondence with the increment in coverage shown in Table 28.

D. Ranking the difficulty of planning tasks

A typical, yet unsolved problem in Automated Planning, is to rank the difficulty of solving planning tasks in practice. Indeed, this problem appeared recurrently in the organization of the seventh IPC and played a major role in the selection of problems to be reused from the previous IPC —see Section 3. This appendix introduces: first, the approach followed to rank the difficulty of solving different planning tasks from the results obtained by a number of planners; second, how this ranking assisted in the task of deciding what planning tasks to reuse from the previous IPC.

A well-known approach to rank participants in two-player games based on their performance is the Glicko rating system [32]. Glicko is widely used to rank chess players but it is also used in many other sports and computer games. Glicko allows to rank two players even if they have not competed directly against each other, by using their results against other players. This is exactly the case of problems in planning: comparisons among problems were performed with regard to their *performance* against planners.

If we regard planners, and also problems, as players each attempt to solve a problem becomes a game so that we can come up with a rank of planners and problems. Each player has a rating, which is a number typically between 0 and 3000. This number results from the outcomes of the previous confrontations with other players. In Glicko, this number is accompanied by a measure of the reliability of the rating, the so called *rating deviation* (RD), which corresponds to the standard deviation of the rating. The RD ranges from 0 to 350. The highest, the less reliable the rating of a player. If r_g is the Glicko score of a player, its actual rating lies in the interval $r_g \pm 2RD$ with a 95% confidence interval. Initially, all the players are given a score of 1500 points and a RD equal to 350. The winner of every contest gains a certain number of points in its rating. The exact number depends both on the difference of ratings between the two players, and also the RDs. The looer, however, does not lose the same amount. In particular, a player is awarded more points by beating a higher-rated player than by beating a lower-rated player, but a lower opponent's RD will make her to earn more points as well. After every game, the RD of both players is reduced, as their rating becomes more accurate. We refer the reader to the original paper for the exact calculations and formulas used to update ratings [32].

Thus, we considered planners and problems of the IPC-2008 as players, and each attempt to solve a problem as a game between them. If a particular planner solved a specific problem it wins the match, otherwise it loses. After every attempt the ratings of both the planner and the problem are updated so that a ranking of the planning tasks used in the last IPC per domain was produced. The idea, however, comes to a price: first, it uses no information about the quality of the solution plans or the effective running time to generate them and, in consequence, two planners solving the same problem would earn a similar amount of points, in spite of the quality of the plans found or the difference in running time; second, the order of the confrontations matter. While nothing was done to overcome the first difficulty, the second issue was addressed by running the algorithm iteratively, confronting every planner with every problem, until ratings converged. Each iteration was considered a different tournament —or *rating period* in the Glicko terminology. Notice that the results of each tournament in terms of win/losses were always the same. Ratings converged usually in a number of iterations between 30 and 50, the exact number being different for each track. Once the Glicko scores were computed, we sorted the IPC-2008 problems according to their rates. Again, let us emphasize that the Glicko score only provides information about how difficult it was to solve a problem by the entrants of the previous competition.

Next we describe the details of the problem selection for each track using the results of the

Glicko score:

- **Sequential optimal track**. The problems of this track were quite challenging as evidenced by the high Glicko scores of a large number of planning tasks. There were many unsolved problems and many planners were not able to solve more than 5 instances per domain. Given the poor performance of many planners in the IPC-2008, instead of generating new problems, many of them were reused. As a general rule, for each domain the n easiest problems —i. e., those with the lowest Glicko score—, were rejected and the following 20 problems, ordered by their scores, were selected. A specific value for n was used for each domain to balance the number of previously solved/unsolved problems (see Appendix B for specific details for each domain). If the number of solved problems was small in a given domain (as for example in TRANSPORT or WOODWORKING), new problems were generated, with the aim of "filling in the gaps" between the IPC-2008 problems.

- **Sequential satisficing track**. In this track a slightly different strategy was followed since more problems were solved. The 10 problems with the highest score were readily selected. Another 10 problems, intended to be more difficult than the previous ones, were generated to complete the 20 problems set. The new problems increase the number of objects and goals.

- **Multi-core track**. To allow direct comparisons, planners in this track were faced with exactly the same planning tasks used in the sequential satisficing track, as mentioned above.

- **Temporal track**. Twenty problems in ascending order of their Glicko score were reused in the most difficult domains. To do that, a procedure similar to the one used in the sequential optimal track was followed. In the easiest domains, the 10 most difficult problems, according to their Glicko score, were reused, and 10 additional ones, expected to be even more difficult, were generated.

E. Statistical tests

Statistical tests are conducted in Section 4 to provide additional evidence of the differences observed in practice with regard to coverage, quality, raw speed and memory usage and in Section 6 to assess the correlation between different metrics. In this Appendix we provide additional information about the way these statistical tests were conducted. For a thorough review of the theory behind these statistical tests we refer the interested reader to a manual on the subject —e. g. [33, 34].

To avoid any assumptions about the distribution of the sampled data, non-parametric statistical tests have been employed. In particular, if data is dichotomic (i. e., each sample is just a measure of only two conditions such as coverage where every instance is *solved* or *unsolved*) the Binomial test [35] is preferred. Otherwise, if data is drawn from a continuous distribution (such as plan quality, the overall running time or the memory used), the Wilcoxon signed-rank test is used instead [36]. Statistical tests performed with regard to raw speed consider only the time to generate the first solution plan even if more solutions are generated. Similarly, memory usage is studied considering the maximum memory ever used and not the amount of memory in use when a solution has been found or at the end of its execution.

Unless stated otherwise, we use 0.001 and 0.005 as the signicance levels because we wish to extrapolate from collections of pairwise comparisons to infer confidence with reasonable probabilities. In the sequential satisficing track (see Section 4.3), which was the largest, 26 pairwise comparisons were performed so that transitivitiy is given with a confidence level equal to $0.999^{26} = 0.974$; in the multi-core and temporal satisficing tracks (see Sections 4.4 and 4.5), which were the shortest, 7 comparisons were required resulting in a similar confidence level in the transitive picture, $0.995^7 = 0.965$.

The Binomial test is an exact two-tailed sign test which measures the deviation of the observed data from the expected distribution that results from assuming that the performance of planner A is the same than the performance of planner B. This distribution corresponds to a binomial distribution with $p = 0.5$. Since it is two-tailed, it returns the p-value that the differences observed between both planners are either too large or too small. As a result, the p-value of each test is compared against a confidence value equal to 99.9% and 99.5%, so that if the p-value returned by the statistical test is larger than 0.001 (or 0.005 respectively), then the Null or Research hypothesis that the two planners being compared perform equally is accepted and the differences are attributed to chance. Otherwise, the Alternate hypothesis that they perform differently is accepted for the given confidence level.

The Wilcoxon signed-rank test is a two-tailed, non-directional statistical test which tests the Null hypothesis that two samples come from the same distribution —and have the same median. Once the differences have been judged as statistically significant, the sum of ranks with positive and negative differences is used to decide what planner dominates the other. In particular, if the rank of the positive differences, R_+, is larger than the rank of the negative differences, R_-, then the first sample is assumed to be significantly smaller than the second with regard to the used confidence level.

The Spearman-rank order correlation test is a non-parametric two-sided test which is applied to *ordinal* data or values that occur in some order of rank. Thus, we are not interested in the difference between any two ordinal values but the differences in the final ranking when following one metric or another. The Null Hypothesis of this test is that there is no correlation between the ranking provided by two different metrics whereas the Alternate Hypothesis suggests that there is a correlation that cannot be attributed to chance. Usual confidence levels for this statistical test

are $\alpha = 0.05$ and $\alpha = 0.01$.

References

[1] C. Linares López, S. Jiménez Celorrio, A. García Olaya, The deterministic part of the seventh international planning competition, Artificial Intelligence (2015). http://dx.doi.org/10.1016/j.artint.2015.01.004.

[2] E. Karpas, C. Domshlak, Cost-optimal planning with landmarks, in: Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09), Pasadena (California), United States, 2009, pp. 1728–1733.

[3] E. Keyder, S. Richter, M. Helmert, Sound and complete landmarks for and/or graphs, in: Proceedings of the Nineteenth European Conference on Artificial Intelligence (ECAI-10), Lisbon, Portugal, 2010, pp. 335–340.

[4] M. Helmert, C. Domshlak, Landmarks, critical paths and abstractions: What's the difference anyway?, in: Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS-09), 2009, pp. 162–169.

[5] C. Domshlak, E. Karpas, S. Markovitch, To max or not to max: Online learning for speeding up optimal planning, in: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10), Atlanta (Georgia), United States, 2010, pp. 1071–1076.

[6] M. Baioletti, A. Milani, V. Poggioni, F. Rossi, Experimental evaluation of pheromone models in ACOPlan, Annals of Mathematics and Artificial Intelligence 62 (2011) 187–217.

[7] R. Fuentetaja, D. Borrajo, C. Linares López, A look-ahead B & B search for cost-based planning, in: P. Meseguer, L. Mandow, R. M. Gasca (Eds.), Proceedings of the Thirteenth Conference of the Spanish Association for Artificial Intelligence (CAEPIA-09), volume 5988 of *Lecture Notes in Computer Science*, Springer, Sevilla, Spain, 2010, pp. 201–211.

[8] C. Candan, J. Dréo, P. Savéant, V. Vidal, Parallel divide-and-evolve: Experiments with OpenMP on a multicore machine, in: 21^{st} Genetic and Evolutionary Computation Conference (GECCO-11), Dublin, Ireland, 2011, pp. 1571–1578.

[9] J. Bibaï, P. Savéant, M. Schoenauer, V. Vidal, An evolutionary metaheuristic based on state decomposition for domain-independent satisficing planning, in: Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS-10), Toronto, Canada, 2010, pp. 18–25.

[10] A. Coles, A. Coles, LPRPG-P: Relaxed plan heuristics for planning with preferences, in: Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS-11), Freiburg, Germany, 2011, pp. 26–33.

[11] J. Rintanen, Heuristics for planning with SAT and expressive action definitions, in: Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS-11), Freiburg, Germany, 2011, pp. 210–217.

[12] J. Rintanen, Planning as satisfiability: Heuristics, Artificial Intelligence 193 (2012) 45–86.

[13] N. Lipovetzky, H. Geffner, Searching for plans with carefully designed probes, in: Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS-11), Freiburg, Germany, 2011, pp. 154–161.

[14] R. Valenzano, H. Nakhost, M. Müller, J. Schaeffer, N. Sturtevant, Arvandherd: Parallel planning with a portfolio, in: Proceedings of the Twentieth European Conference on Artificial Intelligence (ECAI-12), Montpellier, France, 2012, pp. 786–791.

[15] V. Vidal, L. Bordeaux, Y. Hamadi, Adaptive k-parallel best-first search: A simple but efficient algorithm for multi-core domain-independent planning, in: Proceedings of the Third International Symposium on Combinatorial Search (SOCS-10), Stone Mountain (Atlanta), United States, 2010, pp. 100–107.

[16] A. Coles, A. Coles, M. Fox, D. Long, Forward-chaining partial-order planning, in: Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS-10), Toronto, Canada, 2010, pp. 42–49.

[17] F. Bacchus, AIPS 2000 planning competition: The fifth international conference on artificial intelligence planning and scheduling systems, AI Magazine 22 (2001) 47–56.

[18] J. Hoffmann, H. Kautz, C. Gomes, B. Selman, SAT encodings of state-space reachability problems in numeric domains, in: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07), Hyderabad, India, 2007, pp. 1918–1923.

[19] H. Nakhost, J. Hoffmann, M. Müller, Improving local search for resource-constrained planning, in: Proceedings of the Third International Symposium on Combinatorial Search (SOCS-10), Stone Mountain (Atlanta), United States, 2010, pp. 81–82.

[20] 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, Artificial Intelligence 173 (2009) 619–668.

[21] W. Ruml, M. Do, R. Zhou, M. Fromherz, On-line planning and scheduling: An application to controlling modular printers, Journal of Artificial Intelligence Research (JAIR) 40 (2011) 415–468.
[22] M. Helmert, H. Lasinger, The scanalyzer domain: Greenhouse logistics as a planning problem, in: Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS-10), Toronto, Canada, 2010, pp. 234–237.
[23] M. Fryers, M. Greene, Sokoban, Eureka 54 (1996) 25–32.
[24] D. Dor, U. Zwick, SOKOBAN and other motion planning problems, Computational Geometry 13 (1999) 215–228.
[25] J. C. Culberson, Sokoban is PSPACE-complete, Technical Report, University of Alberta, Edmonton, Alberta, Canada, 1997.
[26] B. Marthi, Robust navigation execution by planning in belief space, in: Robotics: Science and Systems, Sydney, Australia, 2012.
[27] D. Bryce, O. Buffet, 6th International Planning Competition: Uncertainty Part, Technical Report, Laboratoire lorrain de recherche en informatique et ses applications (LORIA), Vandoeuvre-les-Nancy Cedex, France, 2008.
[28] D. Smith, J. Frank, W. Cushing, The ANML language, in: The ICAPS-08 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS), 2008.
[29] J. Barreiro, G. Jones, S. Schaffer, Peer-to-peer planning for space mission control, in: Aerospace conference, IEEE, Montana, United States, 2009, pp. 1–9.
[30] D. Long, M. Fox, Exploiting a graphplan framework in temporal planning, in: Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS-03), Trento, Italy, 2003, pp. 51–62.
[31] 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.
[32] M. E. Glickman, Parameter estimation in large dynamic paired comparison experiments, Applied Statistics 48 (1999) 377–394.
[33] S. Siegel, N. J. Castellan, Nonparametric statisics for the behavioral sciences, McGraw-Hill, 1988.
[34] G. W. Corder, D. I. Foreman, Nonparametric Statistics for Non-Statisticians, John Wiley & Sons, New Jersey, United States, 2009.
[35] J. Hoffmann, B. Nebel, The FF planning system: Fast plan generation through heuristic search, Journal of Artificial Intelligence Research (JAIR) 14 (2001) 253–302.
[36] D. Long, M. Fox, The 3rd international planning competition: Results and analysis, Journal of Artificial Intelligence Research (JAIR) 20 (2003) 1–59.