International Planning Competition Uncertainty Part: Benchmarks and Results

Daniel Bryce Olivier Buffet

September 22, 2008

1 Introduction

Organizing an automated planning competition raises various difficulties: deciding of the language requirements, getting people to participate, selecting benchmark problems, finding out which planner is best in the end... This document discusses the last two points in the context of the uncertain part of the International Planning Competition 2008. It also presents the competition results with some comments. A more in-depth analysis would require presenting competing planners in details and is therefore out of the scope of this paper.

2 FOP Track

2.1 Choice of the Benchmark Domains and Problems

Our main objective was to provide competitors with domains with various difficulties and, in each domain, offer problems of increasing complexity.

As usual, most domains can be classified as manipulation domains (blocksworld, exploding-blocksworld), transportation domains (rectangle-tire-world, boxworld, search-and-rescue) and other domains (schedule, sys-admin). Yet this classification is not as important in probabilistic planning as it may be in deterministic planning. In fact, adding probabilistic effects to deterministic domains usually leads to domains which are not *probabilistically interesting* (see [1] for more details on this concept).

So, we decided to look for ways to design probabilistically interesting domains. This led essentially to the two following ideas:

- make sure there are dead-ends and make simple paths to the goal highly likely to lead to dead-ends;
- add action costs or intermediate rewards and make simple paths to the goal less rewarding than more complex ones (note: in previous editions the goal was to maximise the success probability).

Note that the 15 instances of each problem differ not only in their sizes, but also (for certain domains) in their action costs and intermediate rewards. As a result, the difficulty is not monotonically increasing.

Another difficulty is to make sure that the evaluation will be possible in a reasonable amount of time. Some problems (as in the Tetris game used in this year's RL competition) may need long runs and/or many runs to obtain a good performance evaluation.

2.1.1 Blocksworld

This is a modified version of the blocksworld domain used in IPC-5, with:

- some corrections (to avoid ending up with things like "A on B on A") and
- action costs and a goal reward.

The generated problem instances are difficult to compare because the various cost and reward values being used.

2.1.2 Boxworld

This is the same boxworld domain as used in IPC-4 but also with action costs and a goal reward.

2.1.3 Exploding Blocksworld

This is the same exploding blocksworld domain as used in IPC-5 (with new problem instances, though).

2.1.4 Two Tireworlds

In the tireworld domain, a care can move from through networked locations, load a spare tire and replace a flat tire with as spare. The problem is then to maximize the probability to reach a goal location (by going through the safest path). As explained in [1], this is a probabilistically interesting domain, but the instances may be trivial.

To make the problem harder, we have used 1) a specific set of problems and 2) a variant of the domain.

Triangle Tireworld Little and Thiébaux introduced [1] a series of non-trivial tireworld problems that all take the form of a triangle.

Problems p01 to p10 in the "tireworld" subtrack are increasingly complex problems borrowed from Little and Thiébaux.

Rectangle Tireworld One issue with the tireworld domain is that all locations and all connections need to be enumerated, so that generating difficult problem instances requires large PDDL files. We therefore proposed an variant[1] where the network is a rectangular grid, all connections having the same failure probability except on some specific rows, columns or locations.

Problems p11 to p15 in the "tireworld" subtrack are increasingly complex problems from the rectangle tireworld domain.

2.1.5 Schedule

This domain is exactly the same domain as used in the IPC-5, with the same problem instances.

[1] With no flat tires, but a probability of failure for each connection.

2.1.6 Search-and-Rescue

This domain has been proposed by Florent Teichteil-Königsbuch. In this scenario, an autonomous helicopter is on a rescue mission. To achieve its mission, it has to explore several areas to find one that is landable and close to the human to rescue. The original domain [2] has resource constraints that could not be reproduced here.

2.1.7 SysAdmin

The SysAdmin domain was first introduced by Guestrin et al. [3] as an infinite horizon problem: the system administrator has to manage a network of servers who fall down with a higher probability when a neighbouring server is down. The objective is to restart servers one at a time so as to maximize the average number of running servers.

Sanner and Boutilier [4] then proposed a stochastic shortest path version where the system administrator has to go from the "all down" to the "all up" state as fast as possible. Scott Sanner suggested to use this version for the competition.

We have decided to use a stochastic *longest* path[2] version of the problem instead, where one wants to maximize the total number of "hour-servers" before the network is completely down. This is the only domain where planners should try to avoid "goal" states (so that the success probability should tend to be minimized).

2.2 Benchmarking Process

The evaluation process is somewhat different from previous editions, where each team had a 24 hour slot per planner to run it on –possibly all– benchmark problems. With $9 * 15 = 135$ problem instances in 2006, this left little more than 10 minutes of planning+testing time per problem[3]. There was a time limit of 40 minutes, so that competitors had the additional complication of having to distribute the 24 hours among the more or less difficult problem instances.

To get rid of this problem, the 24 hour overall limit was removed. As a result, testing a planner takes about three days if the full 40 minutes are

[2] Many video games enter in this category, e.g. Tetris.

[3] Testing alone takes time because of network communication delays.

used for each problem. To avoid having to set a specific three-days slot to competitors, the planners have been run by the organizers themselves.

Due to time constraints, the time limit had to be shortened to 10 minutes on small problem instances (problems p01 to p05 of each domain). Some tests have been conducted to ensure that this does not significantly change the competition results.

The machine being used to run the FOP track was a quad core CPU[4] at 2.40 GHz with 4GB of RAM and running Linux (Ubuntu 8.04).

2.3 Results

Figure 1 presents statistics gathered on all benchmark problems with all competing planners (+ FF-replan, which was run as a reference planner). There is a subfigure for each domain, with three graphs:

- **%Coverage**: number of test runs ending in a goal state (out of 100); a negative 10 indicates that the planner did not run at all (no connection with the planner);
- **Time (avg)**: average number of time steps before reaching a goal state (when a goal state is reached);
- **Metric (avg)**: per-run average accumulated reward (even if no goal state is reached); this is the performance criterion being optimized.

The first remark to make from these experimental results is that some planners were not able to deal with complex constructs used in some domains (`when`, `forall`...). Part of the competition is then naturally about which planner handles more domains.

The graphs show —as could be expected— that different types of planners are best for different types of problems. Plus, for a given domain, one cannot directly compare problem instances with different reward functions. It is still possible here to point out the "best planners" in each domain as listed in Table 1.

Considering per-domain results (Tab 1) as well as overall results (Fig. 1), RFF-BG and RFF-PG can be declared ex-aequo winners fo the fully observable probabilistic track.

[4]Technical details say: "Intel® Core™2 Quad CPU Q6600".

domain	best planners
blocksworld	RFF-(BG/PG)
boxworld	RFF-BG(/PG)
ex-blocksworld	HMDPP
schedule	SEH
search-and-rescue	FSP-RBH(/RDH)
sysAdmin	RFF-(BG/PG)
triangle tireworld	HMDPP
rectangle tireworld	FSP, RFF-PG

Table 1: Best Planners in each FOP Domain

3 NOND Track

3.1 Choice of the Benchmark Domains and Problems

There already exists quite a few excellent benchmarks for the NOND track from the previous IPC and the literature, and we chose blocksworld (transforming all possible/unobservable blocksworld states into a single state), adder (constructing an n-bit adder from logic gates), and dispose (a grid domain from Palacios and Geffner). We also modified the UTS (Universal Traversal Sequences) problem from the previous IPC to become more difficult (and faithful to the UTS problem). We added two new domains called Rao's keys (a problem with high conformant width) and Forest (unobservable grid navigation with fully observable subproblems).

3.1.1 Blocksworld

The blocksworld domain from the previous IPC was used unchanged. The problems have an initial initial belief state comprised of all legal blocksworld states and the goal is to make a specific stack. The problems increase in difficulty by adding blocks.

3.1.2 Adder

The adder domain comes from the previous IPC. We attempted to make the problems easier to solve by adding extra predicates to describe which bits to the circuit are inputs and outputs (to each full adder) and which bits have been set constant as the result of an action (a logic gate). The intended result of the extra predicates is to cut down the number of actions as well as the branching factor.

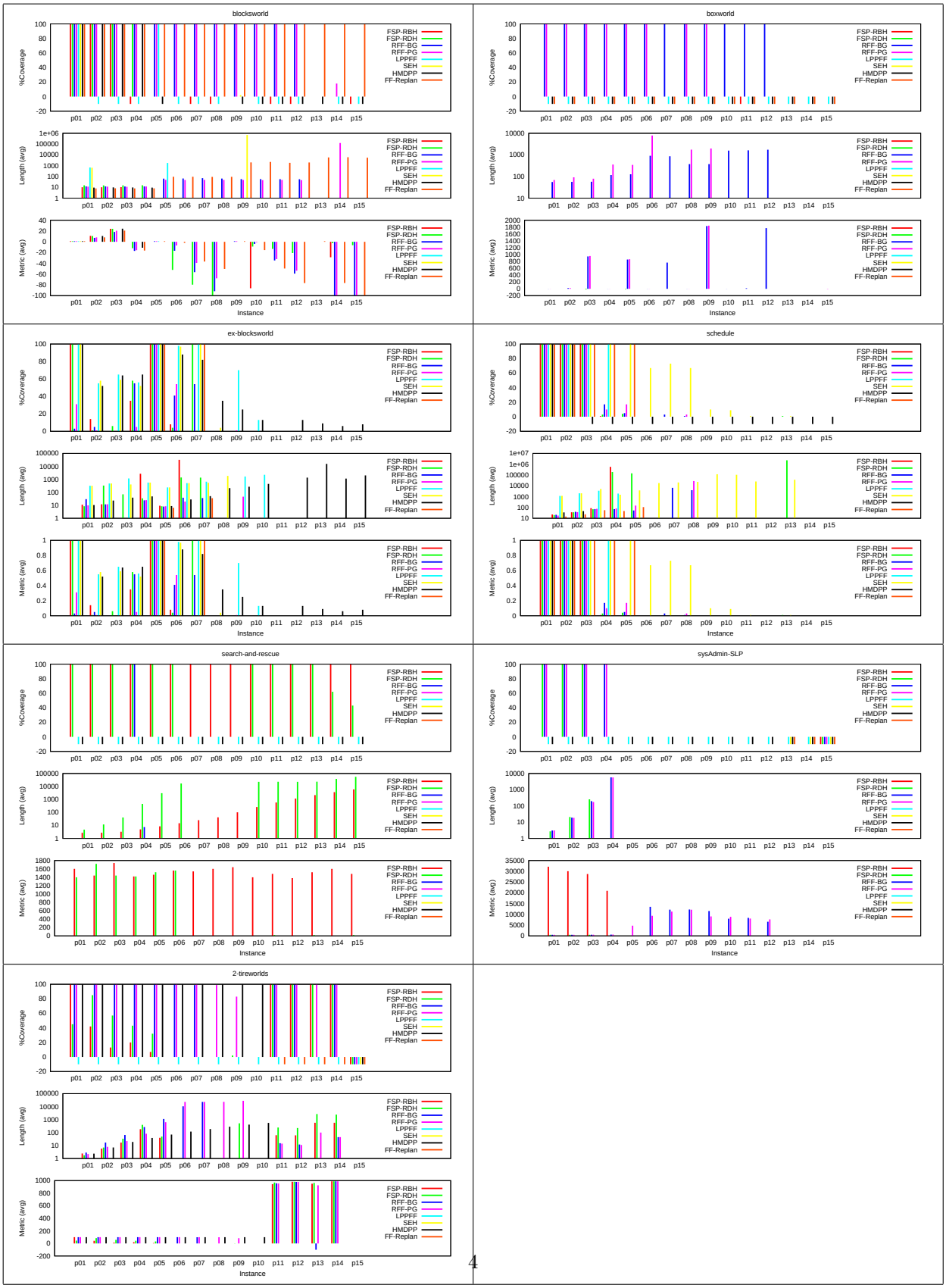

Figure 1: Graphs presenting the results of the FOP track

The problems increase in difficulty by adding bits to the adder circuit (i.e., 1-bit adder, 2-bit adder, etc.).

3.1.3 UTS

The UTS domain was used in the previous IPC, but is somewhat different in this version. The UTS problem is a network routing problem for mobile ad-hoc networks where a broadcast from an unknown node must reach all nodes and the topology of the network is partially known. In the previous IPC, the topology was fully known. Here, each node has a fixed number of connected neighbors and labels (as many labels as neighbors). It is unknown which links lead to which labels; each assignment of labels to links is possible. Moreover, it is unknown which node is currently visited. Thus, plans involve following labels because each node has the same number of links (the graph is regular).

For example, the first instance contains three nodes n1, n2, n3. Each node has two links, labeled either l1 or l2. In node n1 is unknown if following label l1 leads to n2 or n3, and likewise if following link l2 leads to n2 or n3. However, if l1 links to n2, then l2 links to n3, and vice versa. Because it is unknown which node is the originator of the broadcast, each node is a possible starting point. A plan for the first instance is to follow label l1, then label l1, and then label l2 to ensure each node is visited. The problems scale by adding nodes.

We designed two variations, one where each node has degree two, meaning the graph is a cycle. The other variation is also regular and the graph is fully connected, meaning each node's degree is one less than the number of nodes. However, we present results for only the cycle graphs because no planner could solve the fully connected graphs (except of size three, which is also a cycle).

3.1.4 Dispose

We use the dispose domain developed by Palacios and Geffner. The problems involve navigating a known, fully-observable grid to pick up and dispose of various objects in unknown locations. The problems scale by increasing the grid size and number of objects.

3.1.5 Rao's Keys

Rao's keys involves finding a plan for Rao to collect his keys while searching under lights with his sunglasses on. Each key is under a different light and each light is behind a locked gate. The unobservable aspect of the domain is knowing which key opens which gate and which key is located under each light. Rao starts with a single key and it is not known which gate it opens. Plans involve trying each gate, collecting a key, and trying the remaining gates. The problems scale by increasing the number of lights and respective keys.

3.1.6 Forest

The forest domain is actually the template for many domains of a common structure. The structure is a top-level unobservable grid navigation problem with classical planning sub-problems at each grid point. The problem at a grid point must be solved to continue traversing the top level grid. However, because the starting location on the grid is unknown, there is some difficulty in selecting which sub-problem to solve. As in most unobservable grid navigation problems, moving between grid points is an action with conditional effects; each conditional effect requires the subproblem is solved at the corresponding grid point. The problems were generated randomly by assigning different sub-problems to grid points from the following set: a two city logistics problem, the Sussman anomaly blocksworld problem, and a small grid navigation problem (all very easy classical planning problems). In larger problems (larger grids), there is often repetition of sub-problems. A planner using macro-actions would do well on this domain. The problems scale by increasing the size of the grid.

3.2 Benchmarking Process

Each planner was given twenty minutes and 4GB of RAM on the system used for the FOP track for each instance in each domain. We recorded the plan generation time and the plan length for each solved instance.

3.3 Results

Figure 2 presents statistics gathered on all benchmark problems with all competing planners. There

is a subfigure for each domain, with two graphs:

- **Time (ms)**: plan generation time;
- **Length**: the number of plan steps.

Table 2 lists the number of instances solved by each planner in each domain. The declared winner of each domain is bolded and surrounded by a box. The criterion used to select the winner is that planner which solves the most instances, breaking ties in favor of the planner solving instances in the least time. Where the number of instances was equal and solution times were close, the planner finding smaller length plans was declared winner.

domain	CPA(H)	CPA(C)	t0	Problems
blocks	**4**	3	3	4
adder	1	**1**	1	4
UTS Cycle	2	2	**3**	27
forest	1	1	**8**	9
Rao's keys	**2**	2	1	29
dispose	**76**	59	20	90

Table 2: Number of problems solved in NOND Track

The overall winner is CPA(H) because it was winner in the most domains.

4 FOND Track

4.1 Choice of the Benchmark Domains and Problems

The FOND track includes five domains, faults (n-fault planning), first-responders (a logistics-like emergency response domain), blocksworld (adapted from the FOP track), rectangle tireworld (adapted from the FOP track), and forest (adapted from the NOND track).

4.1.1 Faults

The faults domain involves a manufacturing process that consists of several machining operations, each of which could potentially introduce a fault. Each fault can be undone by reversing the faulty operation, but only the most recent fault can be undone. Each problem involves a number of operations and maximum number of allowable faults. Problems scale by increasing the numbers of operations and allowable faults.

4.1.2 First-Responders

The first-responders domain models disaster response situations where a number of fires are present and a number of victims have varying degrees of critical injuries. Fire crews and medical crews can respectively carry water and victims to different locations. Fires are nondeterministically put out by fire crews and victims are nondeterministically stabilized by medical crews in the field. Medical crews can take victims to hospitals to deterministically bring them back to health. Fires, victims, hospitals and water are present at different locations. Problems scale by adding locations, victims, fire crews, medical crews, and fires.

4.1.3 Blocksworld

The blocksworld domain is made nondeterministic by removing probabilities from actions used in the FOP version. All instances used in the competition contain alternative initial states and goals for five block problems. No planner could solve ten or fifteen blocks problems.

4.1.4 Rectangle Tireworld

The rectangle tireworld (described above) is made nondeterministic by removing probabilities from the FOP version. The problems scale by increasing the grid size and number of safe locations.

4.1.5 Forest

The forest domain is similar to the NOND track version, in the following sense. The top-level grid navigation problem is a NOND problem where moving between grip points can nondeterministically result in unintended transitions, and the low-level problem to be solved at each grid point is a classical planning problem. The classical planning problems are identical to those in the NOND version of the domain. The problems scale by increasing the grid size.

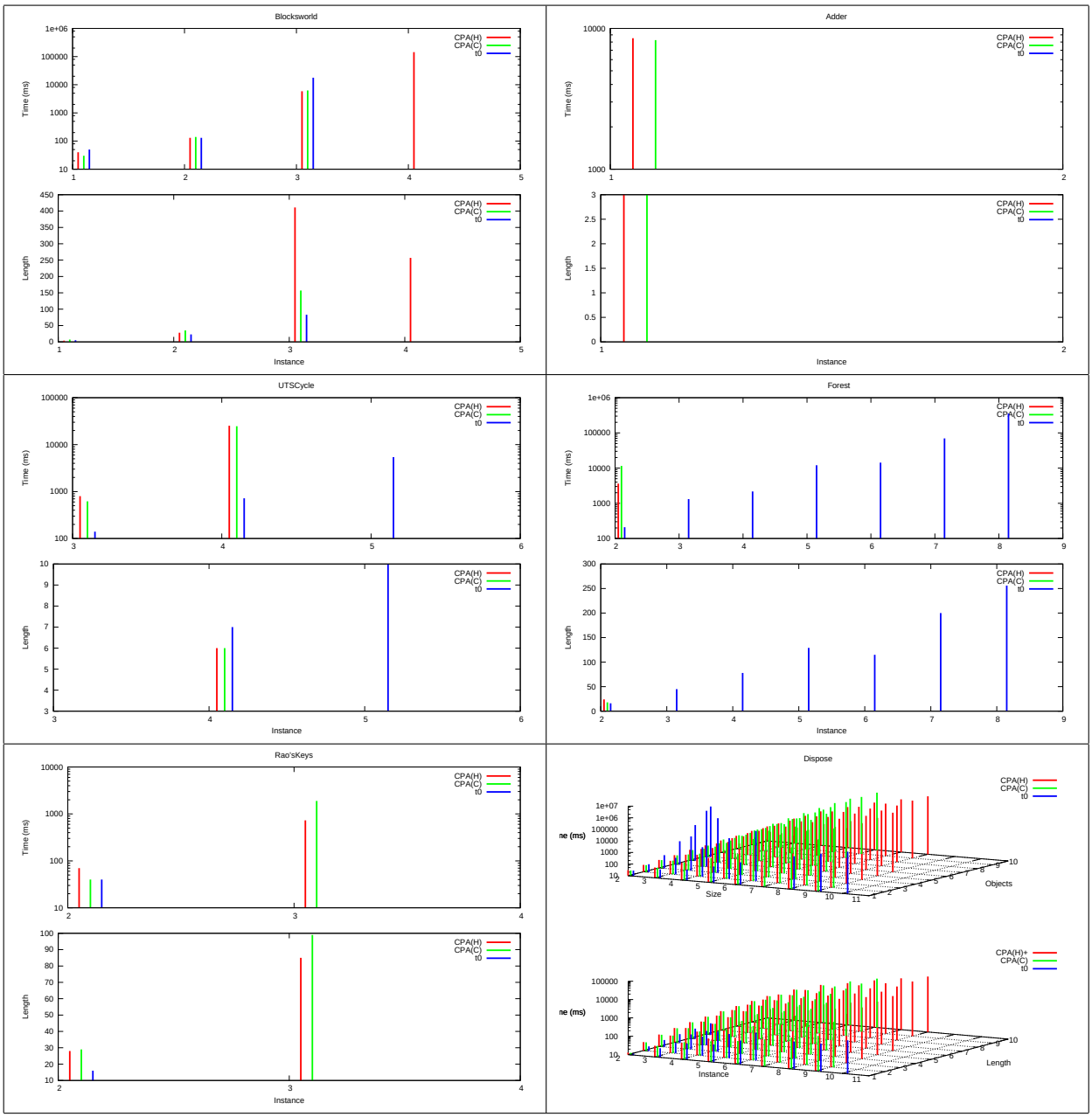

Figure 2: Graphs presenting the results of the NOND track

4.2 Benchmarking Process

We ran the FOND track on a different machine than that used for the FOP and NOND tracks because the track contained single competitor. The machine was an eight core 2.1 GHz Xeon with 8 GB of RAM running Linux. Each instance was given twenty minutes.

4.3 Results

Figure 3 presents statistics gathered on all benchmark problems with all competing planners. There is a subfigure for each domain, with one graph:

- **Time (ms)**: plan generation time;

We award Gamer the best overall performer in the FOND track. While Gamer was the only surviving competitor in the track, we remark that it solved many non-trivial problems with considerably large plans, setting a high-bar for future competitions.

References

[1] I. Little and S. Thiébaux. Probabilistic planning vs replanning. In *Proceedings of the ICAPS'07 Workshop on the International Planning Competition: Past, Present and Future*, 2007.

[2] F. Teichteil-Königsbuch and P. Fabiani. A multi-thread decisional architecture for real-time planning under uncertainty. In *Proceedings of the Third Workshop on Planning and Plan Execution for Real-World Systems*, 2007.

[3] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems (NIPS'94)*, December 2001.

[4] S. Sanner and C. Boutilier. Approximate solution techniques for factored first-order MDPs. In *Proceedings of the Seventeenth Conference on Automated Planning and Scheduling (ICAPS'07)*, 2007.

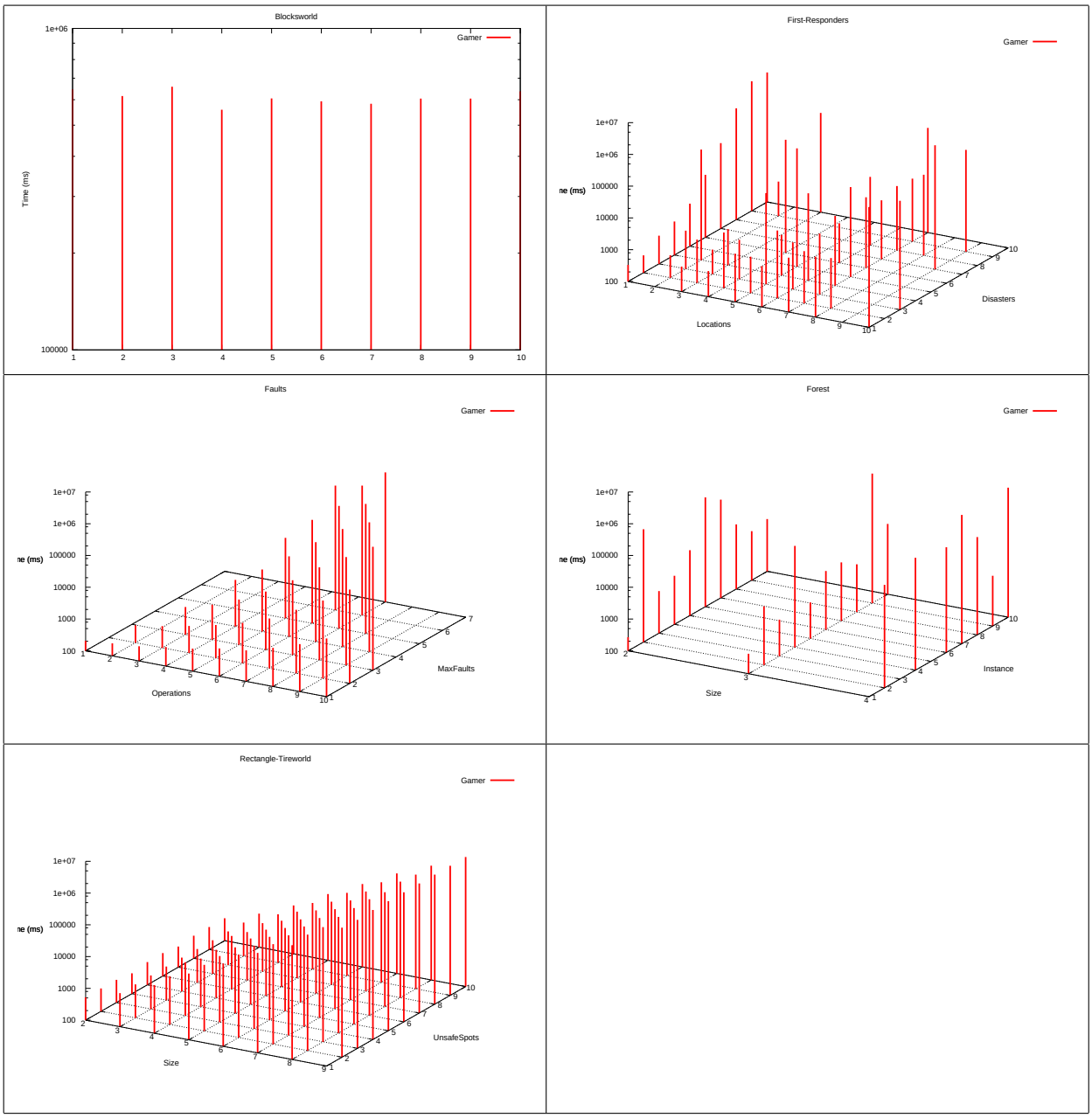

Figure 3: Graphs presenting the results of the NOND track