The Dimensions of Driverlog

Peter Gregory and **Alan Lindsay**
University of Strathclyde
Glasgow
UK
{pg|al}@cis.strath.ac.uk

Abstract

The International Planning Competition has provided a means of comparing the performance of planners. It is supposed to be a driving-force for planning technology. As the competition has advanced, more and more complex domains have been introduced. However, the methods for generating the competition instances are typically simplistic. At best, this means that our planners are not tested on the broad range of problem structures that can be expressed in each of the domains. At worst, it means that some search techniques (such as symmetry-breaking and graph-abstraction) are ineffective for the competition instances.

It is our opinion that a competition with interesting instances (those with varied structural properties) would better drive the community to developing techniques that address real-world issues, and not just solving contrived competition test-cases. Towards this end, we present a preliminary problem generator for the Driverlog domain, and introduce several important qualities (or dimensions) of the domain. The performance of three planners on instances generated by our generator are compared with their performance on the competition instances.

Introduction

The International Planning Competitions have been a driving force for the development of planning technology. Each competition in turn has added to the expressivity of the standard language of AI Planning: PDDL. The domains that have been created for each competition have also increased in complexity and structure. For domains tested in the early planning competitions, such as Blocksworld, problem generation was not considered a difficult problem: generate two random configurations of the blocks and use those as the initial and goal states.

Slaney and Thiebaux showed that even for Blocksworld, problem generation is an interesting problem. Using the intuitive technique to generate states will not generate all possible states (Slaney & Thiébaux 2001). If a simple, intuitive problem generation strategy is not satisfactory for a domain such as Blocksworld, it seems highly unlikely that a similar strategy would be satisfactory for a modern, highly-structured domain.

This work addresses two questions. The first is how to generate an interesting benchmark set for a complex structured domain (the Driverlog domain). The second question asks whether or not the competition results accurately reflect the performance of the competing planners across the benchmark problems that have been created.

Ideally, a set of benchmarks should test current planning technology to its limits. More than simply supplying problems that reach outside of the scope of current planners, a benchmark set should highlight the particular structural properties that planners struggle with. This provides focus for future research. Studying the reasons why our planners fail to solve certain types of problems reveals where future improvements might be made.

Benchmarks should, when appropriate, model reality in a useful way. Of course, it is infeasible to expect planners to solve problems on a massive scale. But it is possible to retain structural features of real-world problems. Nobody would write a logistics instance in which a particular package was in more than one location in the initial state, although this would probably be allowed by the domain file. The structural property that objects cannot occupy more than one location is intuitive, but there may be other real-world structural properties that are not as obvious.

The final function that a good benchmark set should provide is a solid foundation for critical analysis of different planners. One criticism of the IPC could be that there are simply not enough instances to know which planner is best and when. Ideally, there should be enough results to prove that some planner is faster, or produces higher quality plans to a statistically significant level.

The Driverlog Problem

A transportation problem involves a set of trucks moving packages from a starting location to a goal destination in an efficient way. The trucks drive along roads that connect the locations together, and a package can be picked up from or dropped off to a truck's current location. The Driverlog domain extends this model by introducing drivers. Drivers have their own path network that connects the locations together, allowing them to walk between locations. Trucks in Driverlog can only move if they are being driven by a driver. This introduces an enabler role moving away from a simple deliverable/transporter model. As well as this, goal locations are often set for the drivers and the trucks, not just the packages.

Transportation domains can cover problems with interesting road structures. However, Driverlog adds interesting challenges, as there can be complicated interaction between the two graphs structures and there are many more factors to consider when deciding how to deliver the packages, including additional goal types and useful driver truck pairings.

The Dimensions of Driverlog

A Driverlog problem comprises the following things: a set of drivers, a set of trucks, a set of packages and a set of locations. All of the drivers, trucks and packages are initially at a location. A subset of the drivers, trucks and packages have a goal location. Locations are connected in two distinct ways: by paths (which drivers can walk along) and by roads (which trucks can drive along).

We propose eight dimensions that we feel could be combined to create interesting and challenging Driverlog problems. The dimensions largely focus on introducing structural features to the graphs, however, we also consider the types of goals and number of the separate objects in the problem. These could greatly affect the difficulty of the problem.

Graph topology There are several options relating to the graph topology, connectivity and planar or non-planar. Planar graphs are graphs that can be drawn on a plane, with no intersecting edges, a property existing in many real road networks. This domain can be used to represent other problems and it is likely that non-planar graphs will also be of interest and increase the problem space. The connectivity of the graph, from sparse to dense can also be set, allowing a whole range of interesting structures to be explored.

Numbers of objects The number of trucks, drivers, packages and locations are the traditional parameters for generating Driverlog problems. This dimension can be used to set the size of the problem and can have some effect on the difficulty.

Types of goals There are only three possible types of goal in Driverlog: the goal location for trucks, drivers or packages. In real world transportation problems, the planner can never consider delivering the packages in isolation; the final destination of the drivers and trucks is also extremely important. Allowing the types of goals to be selected provides control over the emphasis of the problem.

Disconnected drivers There are two separate graphs in the Driverlog domain, the road graph and the path graph. The interesting interactions that can happen between the two graph structures are usually ignored. We want to encourage exploration of these interactions. Disconnected drivers provide problems where drivers must traverse both graphs (walking, or in a truck) to solve the problem.

One-way streets Links can be added between two locations in a single direction. This means that trucks can move from one location to another, but may have to find a different route to return to the original location. Solving problems with directed graph structure forces careful planning of how the trucks are moved around the structure. If the wrong truck is moved down one of the one-way streets, then many wasted moves could be incurred as the truck traverses back through the graph. As well as adding an interesting level of difficulty, we think this dimension is particularly relevant, because of the increasing number of one-way streets in the transport network.

Dead ends Dead ends are locations, or groups of locations that are joined to the main location structure in one direction only. This means that a truck cannot return once it has moved into one of these groups of locations. This forces the planner to carefully decide when to send the truck into one of these groups, as it will be lost for the remainder of the plan. For example, on difficult terrain there can be craters that a robot can manage to move into, but are too steep for the robot to get out again. In this case the planner may want to balance the importance of the scientific gain with the cost of the robot and termination of the mission.

SAT/ UNSAT This dimension allows the possibility of unsolvable problems. Solvable means that there is a sequence of actions moving the state from the initial state to a state that satisfies the goal formula. This option might allow the exploration of more interesting properties in the other dimensions, as sometimes it is impossible to ensure that certain combinations are solvable.

Symmetry in objects Symmetry occurs in the Driverlog problem when different objects or configurations of objects are repeated. For example, three trucks that have the same start and goal conditions are symmetric. Also, the underlying road network may be symmetric. Planners that perform symmetry breaking can exploit symmetry to reduce the amount of necessary search.

The Instance Generators

Four different generators have been written for this work. In the future, these will be reduced to a single generator. But since this is preliminary work, different generators were produced for different important dimensions. These are Planar, Non-Planar, Dead-ends and Disconnected Drivers. The generators explore the different dimensions identified as interesting in the previous section. Three of these dimensions are not explored: Symmetry in objects, types of goals and SAT/UNSAT. The majority of modern planners have been built around the assumption that instances will be satisfiable, and so this dimension may not produce any interesting discussion. In all of the instances, each driver, truck and package has a goal destination (unless otherwise specified). Symmetry in objects cannot be explicitly varied in any of the generators. It is our intention to add the capacity to vary these dimensions in the future. One more restriction is that except in the Disconnected Drivers generator, the path map is identical to the link map. The generators do the following:

Planar

Generates instances with planar maps. The user can vary the number of drivers, trucks, packages and locations. The user is required to supply the probability of two locations being connected. The user specifies if the map is directed or not. All of the generated maps will be connected. In the implementation, if a generated map is not connected, it is simply discarded and a new one generated.

Non-Planar

The Non-Planar generator is similar to the Planar generator except that the user specifies a particular number of links in the road-map and, of course, the resultant road-maps may not be planar.

Dead-ends

To test road maps with dead-ends, the following method is used for generating an instance. A tree is constructed as the road map randomly, connecting location n with a random location, lower than n. There are t trucks and drivers, initially located at location 1. The last t locations are then used as destination locations for the packages. The trucks do not have a destination location specified.

Each package is randomly assigned a destination from those last t locations. Each package is then initially placed at any location on the path between location 1 and its destination. The challenge in this problem is simply to drive a truck to each destination and only load a truck with packages that are supposed to be delivered to that truck's destination. Figure 1 shows one example. In this example, normal fonts represent the initial location of packages, italicised fonts represent their goal locations.

Disconnected Drivers

The Disconnected Driver generator is designed to explore the Disconnected Driver dimension. In order to do this, a map with no paths is created. Each driver is paired with a truck: the goal locations of the truck and driver are the same. Their initial locations are not the same (although each driver has a truck available). The challenge in the instances generated is in swapping the drivers into the truck that shares its goal location.

Experiments

To test the generators, we have used three of the most successful planners of recent times, FF (Hoffmann & Nebel 2001), LPG (Gerevini & Serina 2002) and SGPlan (Chen, Wah, & Hsu 2006). We used FF version 2.3, LPG version 1.2 and SGPlan version 41. All of the tests were performed on a desktop computer with a dual-core Intel Pentium 4 2.60GHz CPU. The tests were limited to using 10 minutes and 300MB of memory. The timings for FF and LPG measure system time + user time. Sadly, there is no simple way of calculating this measure for SGPlan, and so clock time is used. This could mean that SGPlan seems slightly slower than in reality. However, system load was minimal during testing, and any scaling in performance should be a very small constant factor. The quality of plans is measured by number of actions. As FF only produces sequential plans, and LPG by default optimises number of actions, this was thought a fairer measure than makespan.

We generated a huge number of benchmark test cases, and then after some preliminary small-scale tests chose an interesting selection of problems that covered a range of difficulty for all of the planners. We highly recommend this method. Without preliminary tests, it is impossible to know what range of problems may provide difficulties for the planners. It is far too easy to construct a benchmark set composed entirely of either very easy or impossible to solve problems.

We provide detailed results for planar road maps with four drivers, four trucks, nine packages, and number of locations varying between 10 and 30, in steps of five. For each size of map, we generated 50 instances with probability of two nodes being connected of between 0.1 and 0.9 both for directed and undirected graphs. This gives 250 instances for directed and undirected graphs. Planar graphs were selected as they have a similar structure to real-world road networks.

We also used 180 of the generated Dead End instances. These instances have between one and four trucks, they have 9 packages and all have 15 locations.

Results

The results of performing the above experiments can be seen in Figure 2 to Figure 5. These graphs show the planar directed results (both time and quality) for FF vs. LPG, FF vs. SGPlan and LPG vs. SGPlan respectively. The graphs of the timings are log-scaled, whereas the graphs showing quality are linear scaled.

Time vs. Quality

The results shown in Figure 2 to Figure 5 show that there is little to choose between the planners in terms of plan quality. In each comparison, the two compared planners seem to gain wins in what seems about half of the cases. However, of the three planners, LPG is considerably better in terms of time taken than the other planners. This highlights the fact that planners have been built specifically with the task of attaining satisfiability, rather than trying to optimise metrics.

SGPlan Dependence on FF

It was noticed that in many problems that were found difficult by FF, SGPlan also struggled. FF's performance seems to dominate that of SGPlan. This is unusual, as each goal in a Driverlog problem should be reasonably straightforward to solve in isolation. However, it is perhaps due to the fact that some of the goals in Driverlog are strongly dependent on resources that participate in other goals. This could mean that combining the sub-plans becomes difficult for SGPlan.

Dead-end Example

Figure 1 shows an example of the Dead End instances generated. This instance had three trucks. The numbers in Figure 1 represent package locations. The italicised numbers represent the goal locations of the packages. All three planners were incapable of solving this simple problem.

This highlights the fact that the planners do not reason about resource allocation intelligently. If the problem is viewed as a task of assigning packages to trucks, then the problem becomes very simple. It also shows that the planners do not reason about the consequences of irreversible actions.

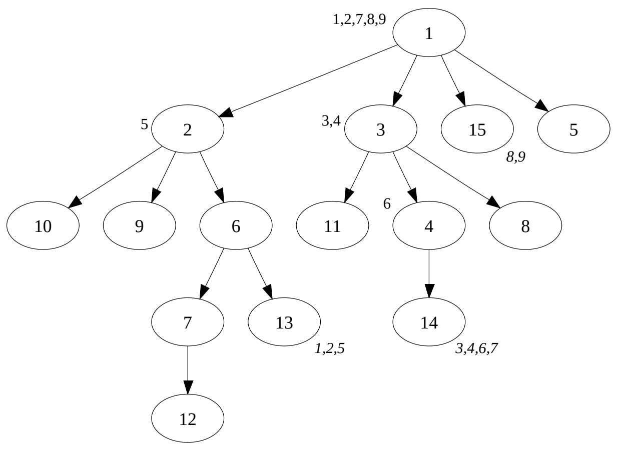


Figure 1: Dead-end instance in which all three planners fail

Instance	1	2	3	4	5	6	7	8	9	10
FF	9	23	13	17	23	14	18	24	32	21
LPG	7	27	13	16	31	14	21	25	32	22
SGPlan	9	24	13	21	24	14	18	30	35	19
Instance	11	12	13	14	15	16	17	18	19	20
FF	26	53	37	39	49	–	–	–	–	–
LPG	25	42	33	78	60	284	143	179	230	176
SGPlan	25	39	36	44	47	–	–	–	–	–

Table 1: Plan Quality for the 2002 IPC Benchmark Instances

Directed vs. Undirected

Figure 2 and Figure 3 show the results of the Planar Directed and Undirected tests respectively. For each of the planners, there was no large difference in the results between the directed and undirected test cases. It was thought that for the same reason the planners deal badly with dead-ends, they may also deal badly with one-way streets. This appears not to be the case, although further experiments may reveal more specific forms of dead-end roads in which the planners struggle.

Competition Comparisons

The planning competition provides a strong motivation in our field and directs the activity of the community. In this study we examined the generator used in the 2002 IPC (Long & Fox 2003): the Driverlog problem generator. We feel that the generator does not provide problems that capture the full potential of what is a structurally rich domain. Therefore it is our opinion that the competition has failed to fully explore how the planners behave in this domain. Our approach focusses on generating problems with several varied structural features and we feel our results provide more understanding of the planners' strengths and weaknesses. We believe that this provides a far stronger base for making comparisons between the planners. In this section we describe the Driverlog generator used in the competitions and discuss the differences between the results of the competition and the results found in this study.

Instance	1	2	3	4	5	6	7	8	9	10
FF	0	0	0	0	0	0	0	0	0	0
LPG	0	0	0	0	0	0	0	0	0	0
SGPlan	0	0	0	0	0	0.1	0	0	0	0
Instance	11	12	13	14	15	16	17	18	19	20
FF	0	0.4	0.2	0.3	0.1	–	–	–	–	–
LPG	0	0.1	0.1	0.2	0.4	85.9	2.8	8.1	52.1	72.1
SGPlan	0	0.2	0.1	0.1	0.1	–	–	–	–	–

Table 2: Execution Time for the 2002 IPC Benchmark Instances

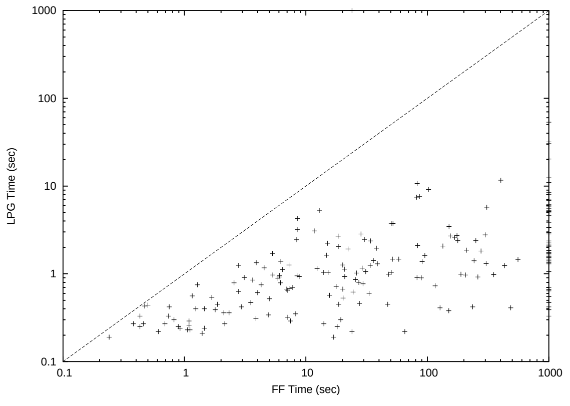
Driverlog is a domain that is rich in structure, however the current competition generator uses a very simple approach to creating the test cases. The parameters to the generator, are the number of trucks, drivers, packages and road locations. The connectivity of the road graph is determined by making (number of road locations $\times$ 4) connections between any two random locations. If the graph is not connected, then additional connections are added between disconnected locations until it is. It is highly likely that this method will produce a very densely connected graph. The same happens for the path graph, except there are (2 $\times$ the number of locations) instead, thus increasing the chances of a sparser path graph. These graphs are both undirected, removing any chances of one way streets or dead-ends and each cover all the road locations, removing the possibility of disconnected drivers. As the graphs are so densely connected it is unlikely that they will be planar and even less likely that they will resemble real-world road networks.

The objects are positioned randomly across the locations and their goal locations (if required) are chosen randomly too. The decision on whether an object has a goal is randomly made, with 95% chance of a package having a goal destination and 70% for both drivers and trucks. This means that no control is given to the types of goal in the problem and no effort is made to position the goals in an interesting way.

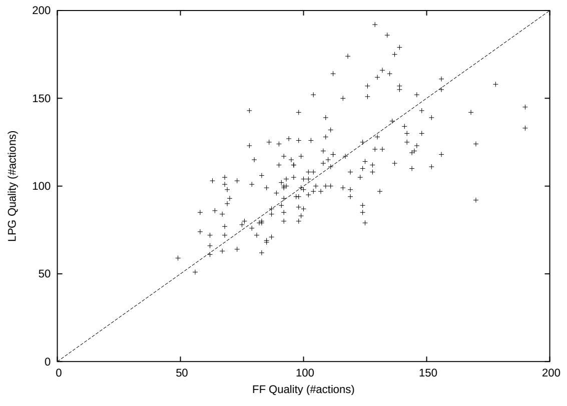
We feel that the planning competition should be able to prove that a planner is faster or produces better quality plans to a statistically significant level. Also, that how a planner performs in a particular area of planning should be identifiable. In our approach we generated problems that incorporated several interesting structural features and spanned a whole range of difficulties. This provides a solid base for judging the performance of the planners across the whole domain and additionally provides invaluable insight into how the planner behaves when faced with specific structural features. We believe that the competition generator fails to explore the interesting features of this domain and makes no attempt to incorporate real-world structures into the problems. Also, we feel that too few problems were generated to determine the performance of the planners. Our results show that our problems spanned a whole range of difficulties, whereas the competition problems were found either too hard or too easy. It is our opinion that the results presented here are sufficient to determine the best planner over the whole domain and in addition, provide useful information to the planner designer, regarding the planner's capabilities.

Depth of results: Number and Range

One of the motivations for this work was to improve the quality of results that the planning competition could pro-

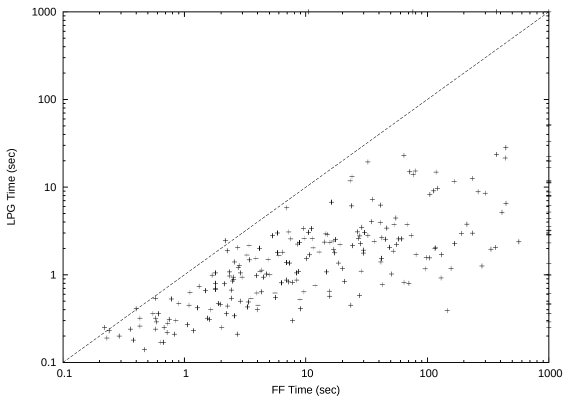


(a) Time

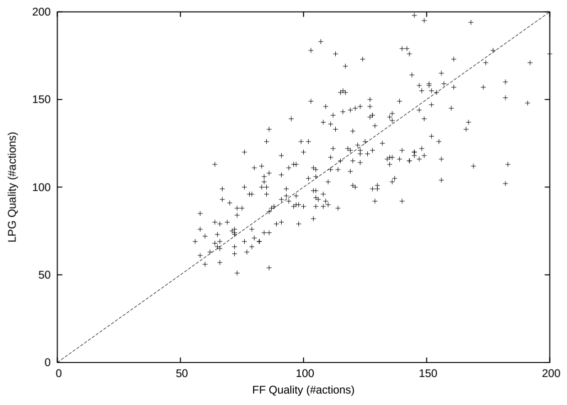


(b) Quality

Figure 2: FF vs. LPG Planar Directed Road Network

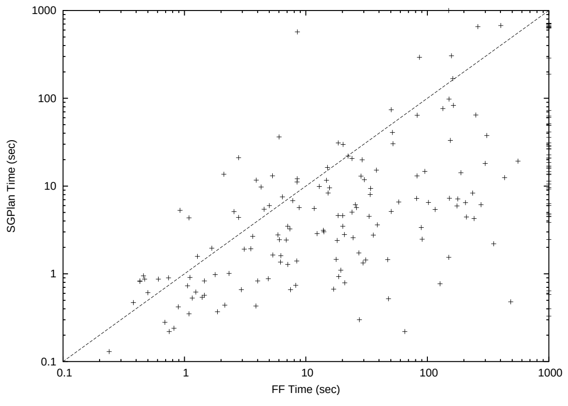


(a) Time

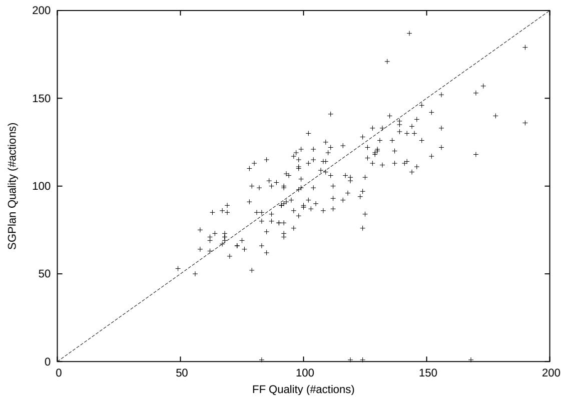


(b) Quality

Figure 3: FF vs. LPG Planar Undirected Road Network

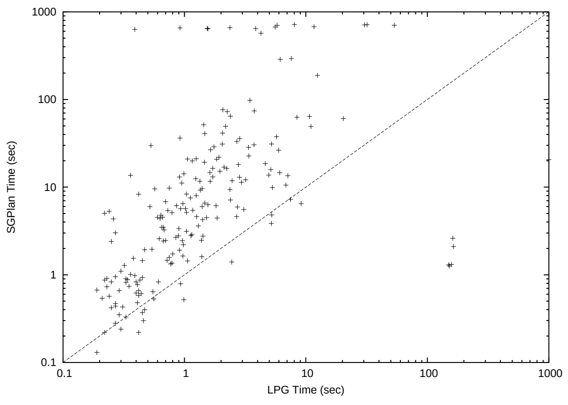


(a) Time

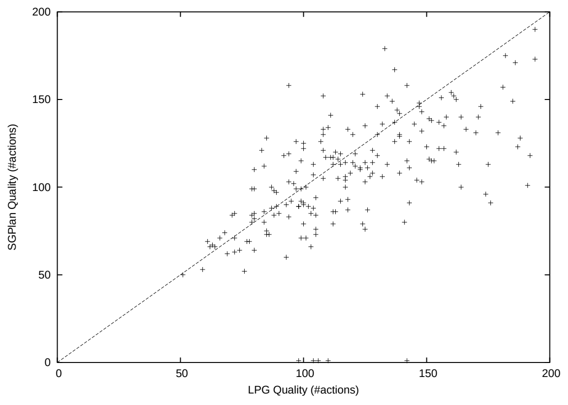


(b) Quality

Figure 4: FF vs. SGPlan Planar Directed Road Network

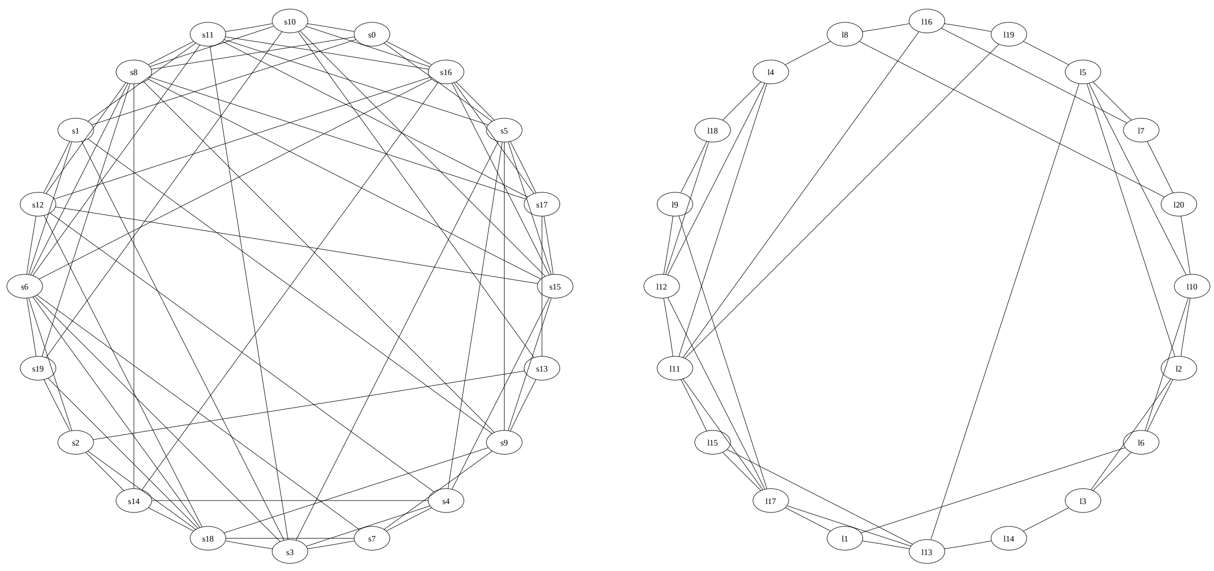


(a) Time



(b) Quality

Figure 5: LPG vs. SGPlan Planar Directed Road Network



(a) Competition (b) Planar Undirected

Figure 6: Competition Benchmark vs. Planar Undirected Graph Density

vide. We feel that the competition would greatly benefit the community if it not only suggested an overall winner, but also highlighted particular features of planning that individual planners excelled in. The Driverlog domain provides an opportunity to test the planners on many interesting structural problems. However, in the competition only 20 problem instances are generated, hardly enough to make a full exploration of the domain. Table 2 shows the time results for FF, LPG and SGPlan for the competition instances. It is difficult to form any kind of of comparison, as the results are so similar. In contrast, Figure 2 b) shows the time result for FF and LPG for our planar problem set. The large problem set ranging over the entire dimension, provides results that clearly shows how the planners compare throughout an entire range of problem difficulties.

The results that we present for each dimension come from a full range of problem difficulties. We feel that this gives us a strong base to make informed claims about each planner's abilities in terms of these dimensions. In the 2002 competition, the first 15 of the problems for Driverlog provided no challenge to the planners, and the last 5 were all found extremely difficult (mostly impossible) (Long & Fox 2003). The problems failed to provide a smooth range of difficulty. We feel that if claims are going to be made about the quality of plans a planner makes or how quickly it produces those plans, then the planner must have been tested across the whole range of possible problems.

Interesting structure

Driverlog problems have the potential of containing all sorts of structural features. We feel that the dimensions introduced earlier, capture a very interesting selection of these. The competition generator constructs the graphs by randomly forming many connections between nodes, and this results in densely connected graphs. All of the graphs are undirected and the road and path graphs must visit every point. This means that the dimensions that we highlighted either can not, or are very unlikely to appear in any of the problems generated for the competition. The competition therefore fails to explore much of the interesting structure possible in this domain.

Our generators cover several structural features; the problems therefore test the planners across these features. This means that our results can be used to determine more than just the best planner: they also identify how a planner performs on problems with a particular structural feature. In the results section, we identified the dead-end feature as a particular problem for FF, LPG and SGPlan. We feel that this sort of information will provide invaluable feedback to the planner designer, allowing them to focus their research on the areas of weak performance. As discussed, it is unlikely that the competition generator will provide many problems with interesting structure. As a result, it is impossible to identify when a planner performs poorly using the competition instances.

Density and realism

The planning competition is a force that directs the planning community and in our opinion it should be used to push planning towards dealing with real-world situations. Although current planners cannot deal with large real-world problems, we feel that realistic structures should be incorporated into planning problems wherever possible. The road connections in real-world transport network often form planar graphs. As we described previously, the competition Driverlog generator is likely to generate very dense graphs, contrasting with the real model. Figure 6 a) highlights the connectivity of a typical competition problem, where b) shows the more realistic, sparse structure generated by our planar graph generator. The dimensions that we have presented in this work, have been designed specifically to test planners on real-world structural features. It is therefore our opinion that our generator is more likely to include realistic structures within the problems it generates.

Future Work

This short study aims to motivate researchers to take the problem of instance generation more seriously. To further this work, several things can be done:

Create More Generators Driverlog is just one domain from many previous competition domains. Instance generators for the full range of competition domains would help to further refine where planning technology's strengths and weaknesses are.

Complete Driverlog Generator Even the Driverlog generators as described in this work are not complete. New interesting dimensions may be identified, which would require extending the generator to create problems across this new dimension. One of the current dimensions (amount of symmetry) is not yet varied explicitly in the generators. Adding this capacity is part of the future work for this project.

Richer Modelling Language PDDL is capable of expressing far more than the propositional instances generated by our current generator. In the IPC, numeric and temporal versions of Driverlog were tested alongside the purely propositional forms of the problem. These included durations on each of the actions, and also fuel costs for driving. They also had different metrics to optimise. Clearly expanding the generators to these dimensions is essential to further planning technology in these areas.

Real-world Derived Instances Real logistics problems are different from typical Driverlog instances both in size and structure. Real logistics problems have huge numbers of locations. The structure of their underlying maps will remain constant: road networks rarely change significantly. If one goal of the planning community is to address real-world problems, then real-world benchmarks are required. Techniques to exploit structures that are constant between different instances could be developed to tackle these problems.

A Generator Generator There are common, repetitive structures that occur in different planning domains. For instance, there are many problems similar to Driverlog, in which movement across a graph is required. If these structures can be identified, then the dimensions identified here that relate to graph structures could be used as generic dimensions in other problems with similar structures. Therefore, if enough different structures could be identified, then a generic problem generator could be created which would be able to generate instances of any domain that have interesting structure.

Conclusions

In this paper, we have tried to show that the problem of instance generation is of critical importance to the planning community. Having complex domains is not enough. To test planners effectively, then benchmarks that explore all possible structural dimensions of our domains have to be created.

We have identified several structural dimensions for the Driverlog domain, and have created instance generators that explore several of these. After creating many instances our results show that, for the planners tested, there is little difference in plan quality. The planners also cannot handle resource allocation intelligently (as seen in the dead-end example).

We have shown that the IPC generator does not generate structurally interesting instances, and have made various criticisms of the competition benchmarks. It must be remembered that running the IPC already requires a great deal of work, and so this work is not created to undermine the efforts of the organisers. However, it does show that creating instance generators should not simply be the responsibility of the competition organisers.

This work is still preliminary, and a completely unified Driverlog generator that can generate instances anywhere in the structural dimensions is essential. There is still plenty work to be done to understand what structural properties underlie difficult instances. Hopefully this work will convince its readers that instance generation is an important topic both for comparing our planners and for understanding what makes a difficult planning problem.

References

Chen, Y.; Wah, B. W.; and Hsu, C. 2006. Temporal Planning using Subgoal Partitioning and Resolution in SGPlan. *Journal of Artificial Intelligence Research* 26:323–369.

Gerevini, A., and Serina, I. 2002. LPG: A Planner Based on Local Search for Planning Graphs with Action Costs. In *AIPS*, 13–22.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*.

Koehler, J. 1999. RIFO within IPP. Technical report, Albert-Ludwigs University at Freiburg.

Long, D., and Fox, M. 2003. The 3rd international planning competition: Results and analysis. *Journal of AI Research* 20:1–59.

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