Experiences with Temporal Planning

Andrew Coles and **Derek Long**
University of Strathclyde
Glasgow, UK

Philip Rendell
SciSys Ltd, Bristol, UK

Abstract

In a recent collaborative project undertaken by the authors, an industrial client expressed an interest in exploiting planning technology to handle a control problem they wanted to solve. The clients had already explored some of the existing planning systems and had a PDDL description of their problem. The systems they had tried had not proved entirely satisfactory and they wanted some help to address the problems. In this paper we describe some of the experiences we had in exploring a version of their planning problem and highlight some of the issues that it raised for the current state of planning technology.

1 Introduction

During summer 2010, colleagues at SciSys Ltd approached the Strathclyde Planning Group, with whom they have collaborated for several years, with a problem brought to them by an industrial client. A slightly simplified and sanitised version of the problem was presented, in PDDL, as a Search and Rescue problem. The problem, described in detail below, is a temporal problem and the client had obtained several publically available planners and tried them. They were not entirely satisfied with the results, for several reasons, and so had approached SciSys because of their ongoing collaboration with the Strathclyde Planning Group and their experience in producing commercial software systems.

In exploring the representation of the problem and performance of temporal planners in solving it, we learned some useful lessons about the way in which current planning technology might confront problems of genuine interest to potential users and, in this paper, we report on our observations and those lessons.

2 The Problem

The structure of the Search and Rescue problem is straightforward: vehicles move between the cells of a grid; at each cell they may conduct a search. The objective is to search all cells of the grid as quickly as possible (ie with a plan of shortest makespan). Two constraints make the problem a little more complex: firstly, each vehicle has a facing (north, south etc) and cannot move in the direction opposing its facing. a vehicle adopts a new facing following a move, which is the direction it moved to arrive at the new cell. Secondly, vehicle movement and search are durative actions, with different durations. The client chose to model movement actions with durations dependent on a speed value associated with the moving vehicle. The vehicles are modelled with fuel which is consumed by movement. A vehicle can only move if it has sufficient fuel to do so.

In the original encoding of the problem presented to us, the search grid was represented by a coordinate space, with each location being denoted by a pair of objects (the x- and y-coordinates). We proposed a change to single names for locations, which simplifies the action encoding and reduces the number of grounded action instances that has to be considered during initial processing, but makes the encoding of larger problem instances slightly more tedious (the number of location objects and associated linking predicates increases quadratically instead of linearly). In this paper we restrict attention to this version. The client chose to encode facing and position of vehicles with two separate predicates. This is a natural encoding, but, as we discuss later, it turns out that an encoding with a single predicate has advantages. The client also began with encoding in which the constraint on facing for movement was captured as a disjunctive precondition for the move action. This severely constrained the range of applicable planners, so we proposed a modification in which the vehicle facing is constrained to be unequal to a banned direction for the link between cells used in a particular move action. This encoding is slightly less immediately intuitive, but is important in granting access to a wider pool of planners.

The encoding for the actions in the domain is given in the appendix. Problem instances are described in detail below. We considered problems with a range of different grid sizes (5x5 to 10x10), combined with different numbers of vehicles (1–6).

It is worth noting that this problem is a variant of the Open Vehicle Routing Problem (and the Travelling Salesman Problem). This problem has been the subject of considerable research effort: a good overview can be found in (Li, Golden, and Wasil 2007). The best results for this problem appear to have been generated by tabu-list metaheuristic approaches, using domain-specific heuristics such as node clustering. Results are reported in (Li, Golden, and Wasil 2007) for several approaches, solving instances up to 200 nodes with as many as 16 vehicles. One of the better solvers is reported to solve a 50 node instance, with 5 vehicles, in 0.22 seconds. This datapoint helps to set in context the performance of the planners we explore, which are generic solvers not tuned to solve this problem.

3 Initial Experience

The client had experimented with SGPlan5.2.1 (Chen, Wah, and Hsu 2006) (note that this paper describes the first version of SGPlan and SGPlan5.2.1 appears, in fact, to behave rather differently to this description — we give some insights into its behaviour below, based on our exploration of its code base). This planner can solve the problems reasonably quickly and the performance was considered promising for the application being developed. However, two problems were of concern. Firstly, the code base release has a licensing status that is not entirely clear (it uses some of the FF code base (Hoffmann and Nebel 2001), which is released with a GPL licence, but SGPlan itself does not appear to adhere, unambiguously, to the terms of that licence). In any case, routes to continue development or debugging of SGPlan are not clear. Secondly, the quality of the plans generated by SGPlan is poor. We explore this problem further, below.

Once we had modified the encoding as described above, we were able to test a variety of other temporal planners. These included Sapa (Do and Kambhampati 2003), LPG-td (Gerevini, Saetti, and Serina 2006), Temporal Fast Downward (Eyerich, Mattmüller, and Röger 2009), CRIKEY3 (Coles et al. 2008) and POPF (Coles et al. 2009). MIPS-XXL (Edelkamp and Jabbar 2006) failed on the domain after having constructed merged automata. VHPOP (Younes and Simmons 2003) does not handle the numeric parts of the domain and the newer temporal and metric LPG (Gerevini, Saetti, and Serina 2010) proved to be less robust than the earlier version. We do not believe that the modifications in the new LPG would make significant (beneficial) differences to its performance on this problem, since they are intended to allow the planner to solve problems with required concurrency: although this problem benefits from exploitation of concurrency, there is no required concurrency.

Our initial experiments, with 5x5 grids, indicated that most of these planners produce plans of poor quality. This problem arises from a key weakness in the decomposition of temporal planning problems into the selection of actions and then subsequent scheduling. This decomposition typically ignores the temporal costs of actions as they are considered for inclusion in the plan (or, at least, the implied future costs following the commitment being evaluated). Most of these planners select actions based on the number of actions they estimate will then lead to a plan. In some cases, this choice is further informed by the cost of the action being considered, but rarely does the heuristic actually take into account the expected makespan of the subsequent plan. In part, this is because the relaxed reachability graph in which the heuristic evaluation of states is performed makes it difficult to make useful assessments of temporal costs. Furthermore, the construction of the heuristic values depend on the construction of a relaxed plan and it is important that this construction be fast enough to perform multiple times at every search node. This entails that the construction does not rely on search and this makes the identification of temporally efficient relaxed plans particularly difficult.

A consequence of this is that it is common for a temporal planner to select actions that must, in fact, be sequenced, due to use of common resources, when concurrent activity is possible using different resources. For example, in this Search and Rescue problem, when there are multiple vehicles available, it is clearly better to use all of the vehicles (if they are equivalent, then one would hope to see an even share of tasks between them). However, it is often the case that planners will use the vehicles very unevenly — sometimes using one vehicle to perform all, or almost all, of the search.

3.1 SGPlan 5.2.1

SGPlan adds a further decomposition to the planning-scheduling split: it breaks the goals into subsets of 5 at a time. It proceeds to solve the first set from the initial state, using a version of FF, before moving on to consider the next 5 goals, starting from the state that it has reached after the first set. This process iterates, with the constraint that each successive sub-plan must preserve the goals achieved in the preceding part of the plan. In cases where this constraint is violated, the planner explores ways to reachieve the violated goals while maintaining the other goals. This repair strategy extends over several iterations before search returns to consider larger subsets of the goals. In our problem this machinery is not relevant, since the goals can never be undone by subsequent activity in the plan. Thus, SGPlan reduces, for our problem, to an iterated application of FF to sets of 5 goals at a time.

In fact, this approach offers some benefits for this problem: the successive sub-problems are constrained in size, so the planner can solve relatively large instances surprisingly quickly. In problems where there is only one vehicle, provided that the goals are considered in a good order then the plans will be very efficient as well as being produced extremely quickly. The subsets of goals are not constructed intelligently, but simply taken from the list of goals as they are stored in the internal data structure of the planner. This means that the order goals are considered depends on how this data structure orders them, which turns out to be dependent on the order in which they are parsed. It is natural to present the goals in a logical order when constructing problem instances, but experiments with randomised orderings show that SGPlan is highly sensitive to the problem description. In contrast, the other planners are not affected by the order goals are presented. Furthermore, the decision to split goals into sets of 5 is an arbitrary one. It happens to work quite well with 5x5 grids, leading to good plans (when the goals are also presented in a logical order), but it works less well when the grid is 8x8, for instance. In other domains, treating goals in groups of 5 can lead to extremely poor behaviour.

4 Comparing Temporal Planners

The Search and Rescue problem is an interesting one for demonstrating several aspects of temporal planning in a context in which there is no required concurrency. The optimal solution must share the tasks between the vehicle resources, with the precise share depending on the relative capabilities of the vehicles (both available fuel and speed for each vehicle). It must also plan the paths of the vehicles very carefully, so that they do not interact in ways that leave unexplored regions that can only be visited by revisiting locations that have already been explored. Good quality solutions will similarly exploit multiple vehicles and attempt to minimise revisiting, but a general solution to this problem is NP-hard, so it is likely that a heuristic solution will often find sub-optimal plans.

The fact that concurrency is clearly key in achieving a good sharing of tasks between resources led us to expect that planners that reason about concurrent activity directly in the heuristic computation should produce better quality plans. On the other hand, doing so is likely to be more expensive, so we expected that planners that simplify the reasoning by using the technique of action compression (treating actions as instantaneous) and ignoring concurrency, until a post-processing scheduling phase, should generate plans more quickly.

The results for a 5x5 grid with two equivalent vehicles starting at opposite corners are shown in Table 1. An optimal plan for this problem should use one vehicle to search 12 locations, the other for 11 (the starting locations are considered to be already searched). The duration for a search of 12 locations will be 36 (plus the necessary separations between the movement and search actions to avoid interactions — since planners handle this demand slightly differently, we will abstract it throughout our discussions and presentation of results). The following points should be noted in reading the table: the first values for SGPlan use a logical ordering of goals, while the second collection uses a random ordering. The ranges reported for LPG-td are based on 5 runs producing 4 solutions in each case. The best solution of the 20 generated has makespan 42 with 53 actions. The two parts of the time reported for Sapa are grounding and search respectively. It is clear that a more efficient grounding procedure could be implemented. The plan reported for TFD is the 3rd plan it produces (which is the best it produces over a more extended run).

It should be clear, from this table, that the approach taken by SGPlan is fast, but produces poor quality plans. In contrast, Sapa produces good plans, but takes much longer to do so. It is also clear that none of the planners produces an optimal plan, but the plans produced by Sapa and POPF are both good quality.

These results also tend to confirm our expectations, while revealing that the temporal reasoning in CRIKEY3 is not as powerful as we might hope. It turns out that the extensions introduced in POPF are extremely powerful in this problem. POPF extends CRIKEY3 by constructing the plan as a partially-ordered structure, adding total ordering constraints only when interactions in the plan require it. This allows POPF to delay commitment to some of the ordering constraints. In this problem this is very helpful, because the ordering constraints are only required between actions that use the same vehicle: where actions involving different vehicles are interleaved, the opportunity to avoid enforcing an arbitrary ordering at the point of action selection is very powerful. Sapa, CRIKEY3 and POPF use some temporal information in the reachability graph, allowing them to attempt to minimise makespan during the planning process.

It is worth noting that POPF can be run with several different configurations. In this problem we found that some of the ideas first applied in FF offered significant benefits. These include ordering helpful actions to prefer those that do not delete the preconditions of other helpful actions (which favours performing a search before moving a vehicle away from a location) and also, during relaxed plan extraction, preferring actions that have lower cost under the h_{add} heuristic for achieving preconditions. We also tried ordering actions so that those that could be applied earlier were preferred, but this turned out not to be useful in this domain. The use of these additions to the heuristic made as much as an order of magnitude difference in the performance of the planner on this problem, while also helping to improve the quality. All of the results we report are with these features enabled.

LPG-td-1.0 uses a local-search based approach. This is very closely related to the approaches that have proved to be most successful in the Vehicle Routing Problem literature. In particular, the use of random restarts is a powerful metaheuristic for tackling heavy-tailed distribution problems (problems where there is high proportion of poor quality solutions). As will be seen in later results, LPG-td performs well in producing good quality solutions. However, it has a high variability in its performance. To achieve a good quality solution requires that the planner be run over much longer periods than the client was prepared to wait (a few seconds, at most), while the variability and unpredictability of its behaviour made it very much less attractive than the deterministic planners.

4.1 Scaling Behaviour

A 5x5 grid is rather small, but it quickly became apparent that the scaling behaviour of our temporal planners is a limiting factor in the applicability of the technology for more complex scenarios. Two problems apply, here: firstly, the grounding of actions becomes much more expensive as the grid size grows. A quadratic growth is to be expected, but in fact it can be worse than this, depending on the strategy used for grounding, because the move action contains two location parameters. This can lead to a quartic cost increase in the width of the grid if the grounding approach is too naïve. Secondly, the length of plans grows quadratically in the width of the grid, but growth in plan length can have a super-linear impact on the cost of construction, depending on the accuracy of the heuristic. For a challenging problem, increasing plan length could have an exponentially growing time cost for its construction if the heuristic is poorly informed.

To explore the scaling behaviour as the grid size increased, we considered the performance of our collection

Planner	SGPlan5.2.1		LPG-td	Sapa	TFD	CRIKEY3	POPF
Time (s)	0.07	0.1	0.72-7.97	40.8+1.2	8.4	1.05	0.36
Makespan	60	80	42-45	39	52	55	39
Number of actions	62	79	52-54	50	57	51	50

Table 1: Results for searching a 5x5 grid with two vehicles. The two columns for SGPlan are for logically ordered goals and for randomly ordered goals, respectively. The results for LPG-td indicate the range of performance. The time reported for Sapa shows the time spent grounding the problem and the time spent searching, separately.

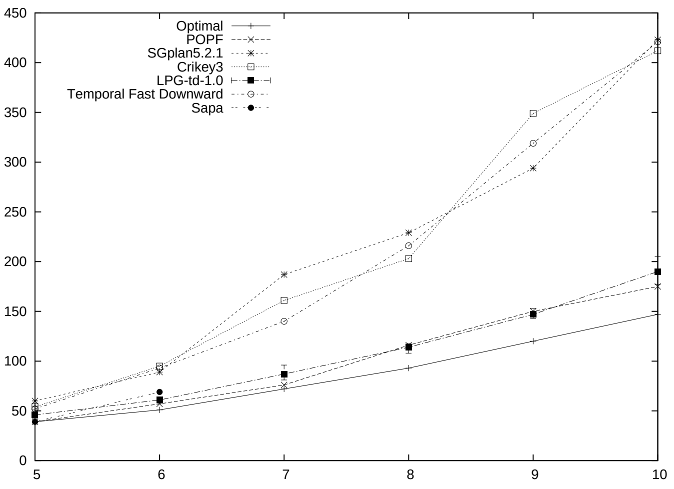

Figure 1: Makespan of solutions produced by a range of temporal planners on two-vehicle problems with varying size grids. Optimal plan makespan shown for comparison.

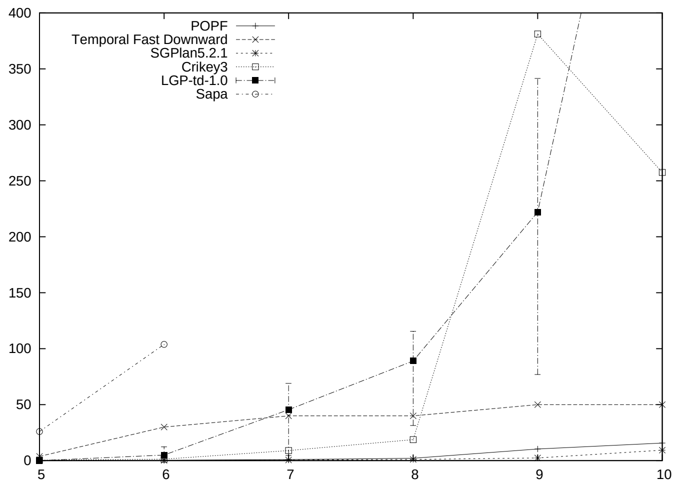

Figure 2: Time (in seconds) taken to find solutions by a range of temporal planners on two-vehicle problems with varying size grids.

of planners on grid sizes between 5x5 and 10x10. In these problems we continue to use two vehicles. Scaling behaviour with an increase in the number of vehicles is more difficult to assess, since increasing the number of vehicles should decrease the duration of the plan, but generally not the number of actions it requires, while it also increases the number of grounded actions (linearly in the number of vehicles). We examine this behaviour below.

The makespans of the solutions is shown in Figure 1 and the times are shown in Figure 2. Sapa only solved problems of size 5 and 6 within 1800 seconds. The time to ground the problems dominated for these sizes, but ceased to dominate by size 7. The range of values for LPG-td was quite large, particularly for time. The error bars indicate the range of values for 5 runs using the quality setting. In the case of the timing graph, the value for the upper end of the range for size 10 problems is around 1800 seconds.

We now consider the scaling behaviour with respect to vehicles. We work entirely within 6x6 grids, to focus on the effects of the vehicle numbers. The makespans are shown in Figure 3 and the times for these plans in Figure 4. To avoid compressing the second graph, we cut off the upper end of the LPG time range, which is at 23.01 seconds. LPG-td has a widely varying performance: the best plans are generally the ones that it takes longest to produce and, conversely, the worst quality plans are produced fastest. Sapa failed to solve any problem in under 25 seconds (it could solve all but the one-vehicle case within 90 seconds, but the grounding dominates the search by more than an order of magnitude in these problems). Temporal Fast Downward requires 30–60 seconds to solve these problems with the plan qualities reported.

Figure 3 shows that all of the planners respond to the increased opportunity for concurrency. In order to clarify the relationship between these plans and the nominal optimal plan, we also show the relative length of the solutions generated compared with optimal in Figure 5 (we did not construct the optimal solutions, so these are based on the assumption that each instance can be solved by an equal distribution of tasks between the vehicles). In this figure we use a representative solution from LPG-td. In general, LPG-td, Sapa and POPF produce plans within 40% of optimal, or better, although all of these planners perform worst as the opportunities for concurrency are greatest. This is partly because, as the number of vehicles increases, the grid becomes more crowded, so that the segmentation of the space between the vehicles becomes significantly harder.

4.2 Temporal Sensitivity

We now consider the extent to which these planners are able to perform integrated planning and scheduling reasoning. We consider the effects of changing the relative speeds of

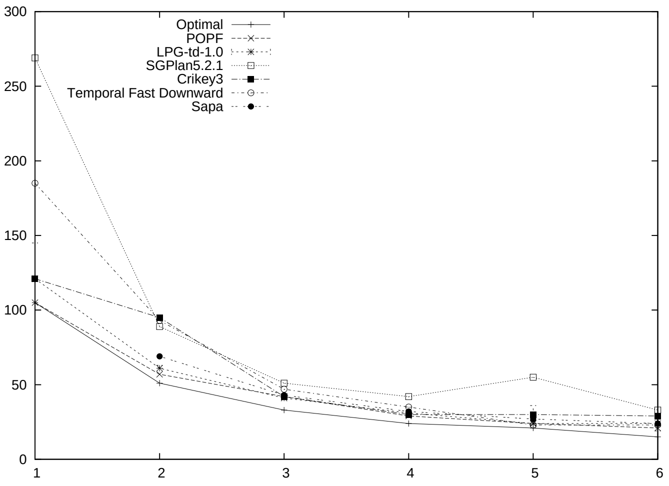

Figure 3: Makespans of solutions on 6x6 grid with varying number of vehicles.

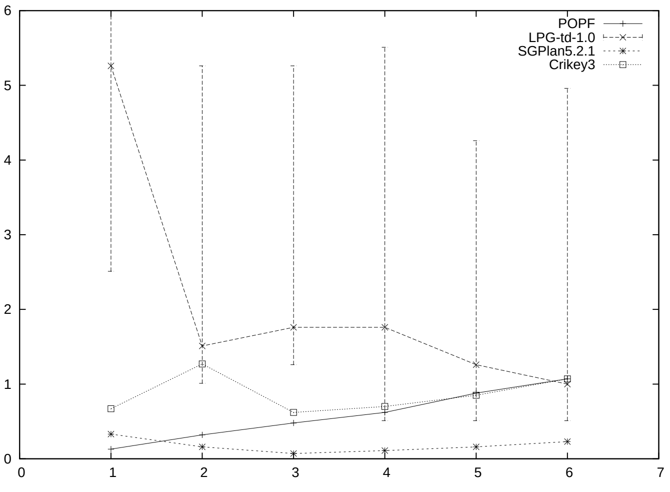

Figure 4: Time (in seconds) taken to find solutions by a range of temporal planners on 6x6 grids with varying number of vehicles.

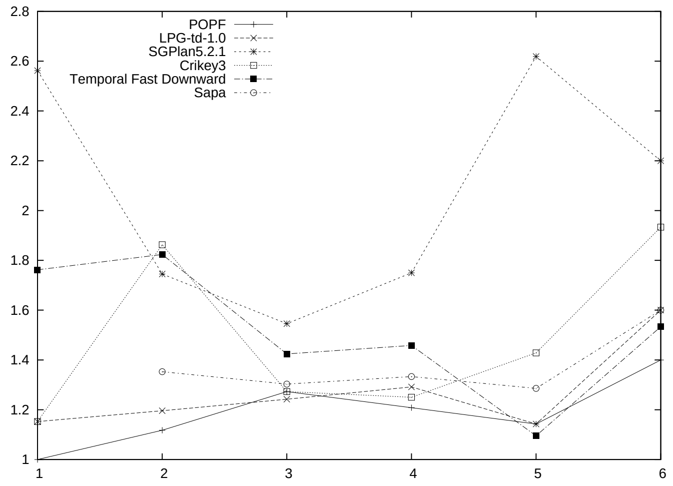

Figure 5: Makespans of solutions for 6x6 problems using different vehicle numbers, as proportion of nominal optimal plans.

two vehicles used to search a 6x6 grid. If a planner fully decomposes planning and scheduling then we would expect to see that changing the relative speeds will have no impact on the plan and the subsequent scheduling will be unable to improve the plan quality, while a more integrated reasoning will allow the planner to select actions that support a better schedule during plan construction, leading to higher quality plans.

The results, shown in Figure 6, indicate that POPF and LPG-td are both sensitive to changing relative speeds. Results for LPG-td use the best of 5 runs. Inspection reveals that SGPlan generates identical plans in all cases, so is completely insensitive to the changing problem specification (hence generates linear increase in the makespan with increasing relative slowness of the second vehicle). Sapa cannot solve these problem instances fast enough to be interesting. Temporal Fast Downward could not solve the problem instance in which one vehicle was 3 times slower than the other, due to a bug, but the trend in the other results is clear.

4.3 Time and Resources

A further constraint on the structure of plans can be introduced by reducing the fuel-efficiency of one or other of the vehicles. As an example that illustrates the way this can affect plans, we consider a slow vehicle (10 times slower than the other) and a fast, but less fuel-efficient, vehicle. By varying the fuel-efficiency, we can modify the number of locations that the faster vehicle can visit. This situation has an interesting impact on the search space: clearly it is better to use the faster vehicle to do as much as possible, while the slow vehicle must do the rest. For a forward-search planner, that commits quickly to its decisions, this is not particularly problematic. For a local-search based planner, this problem is much harder: the interleaving of the few moves of the faster vehicle with the many moves of the slower ve-

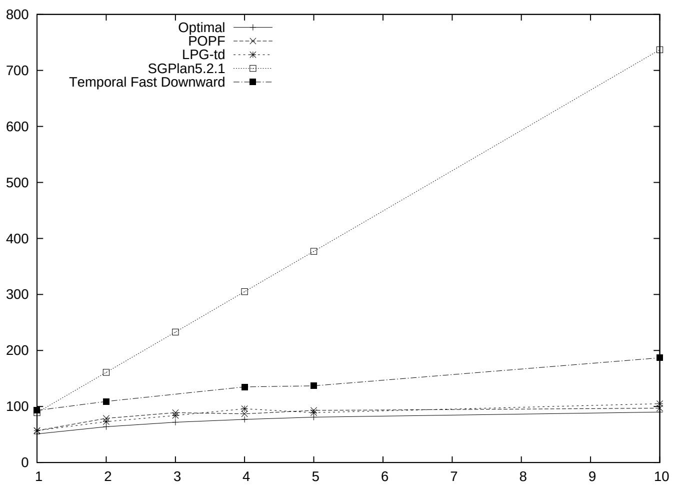

Figure 6: Makespans of solutions for 6x6 problems using two vehicles, with one running at the indicated (x-axis) relative speed slower than the other. Optimal makespans are nominal, since optimal plans were not constructed.

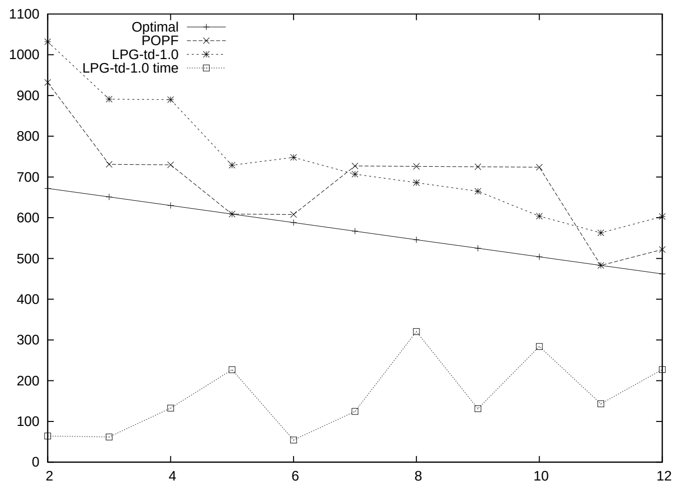

Figure 7: Makespans of solutions for 6x6 problems using two vehicles, one 10 times slower than the other, and with the fast one constrained to a varying maximum number of moves (x-axis). The execution time of LPG-td is also shown (in seconds). POPF solves all problems in under 0.5 seconds.

hicle creates considerable symmetry structure in the problem (plan permutation symmetry) and this makes the search space huge without any benefits.

The results, shown in Figure 7, indicate that LPG-td finds this collection of problems hard to solve, while POPF continues to solve them quickly (its time performance is under half a second for each of these instances). However, the quality of these solutions is significantly worse than in previous examples. The reason for this is that POPF does not plan an optimal path for the slower, but more active, vehicle and the penalty for the unnecessary extra movement of the slow vehicle is very high in this problem (20 time units per move). The gap between the (nominal) optimal solution cost and the solution found by POPF shows that it adds about 10-12 extra moves to the length of the path for the slower vehicle in the worst cases, although in some cases it manages to achieve far better performance. Due to the slow performance of LPG on these problems, the figures are based on a single execution, so should be considered only representative of the performance. Nevertheless, the execution times are consistently 2-3 orders of magnitude slower than POPF.

4.4 Routing

As a final brief examination of the relative performance of some of these planners, we consider the task of routing a single vehicle. In this case, the problem becomes a variant of a simple Travelling Salesman Problem, with no requirement to return to the starting point, but with a constraint that the vehicle can never turn immediately back on itself. This constraint should not impact on the planners. We are interested in the quality of the tour each planner can produce. We only consider LPG-td, SGPlan5.2.1 and POPF in this case. In Figure 8 can be seen the routes proposed by POPF and LPG-td: the former generates the optimal lawnmower pattern, while the latter produces a plan that tracks locally around the edge

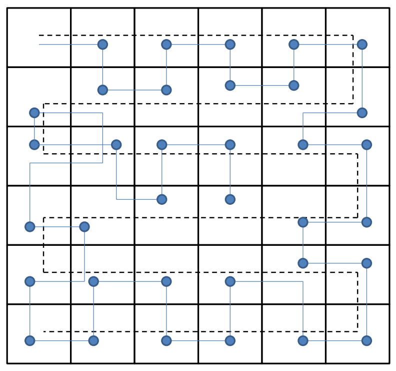

Figure 8: Search paths for single vehicles on 6x6 grids, produced by POPF (dotted path) and LPG-td (solid path with dots showing when searches are conducted).

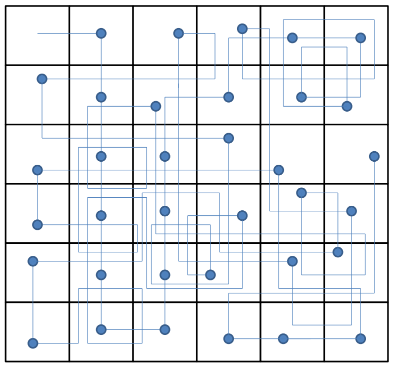

Figure 9: Search path produced for SGPlan5.2.1 for a single vehicle searching a 6x6 grid. Dots indicate when searches are conducted.

and then proceeds to the centre, but with some inefficiency (this was the best plan using a '-n 10' setting, which took several minutes to run — the plan has a makespan of 119 compared with the optimal 105). It is clear that local search might have some difficulty in improving this plan without using a carefully crafted neighbourhood for the search to enable intelligent restructuring of the path. Figure 9 shows the plan generated by SGPlan (with a makespan of 269). Note that the goals are given in a logical order. The strange spiralling revisiting of locations, particularly around the centre of the grid, appears to be a consequence of the facts that the centre is an attractive location for the relaxed plan to visit (giving best access to the rest of the grid) and that SGPlan considers the goals in small sets, which limits its ability to position itself well for the next sequence of search locations.

5 Variations on a Theme

One of the benefits to the client of using planning technology, rather than a dedicated solver, is the flexibility to easily modify the structure of the domain and still (often) get good performance from the solver. Indeed, the problem the client was interested in solving was not exposed to us at all, but was simply described as 'similar to the Search and Rescue domain'. As a final experiment, we therefore explore the flexibility of the planners.

One variant of interest is to switch the goals from exploration of locations to exploration of edges between locations. This is analogous to a switch between the Travelling Salesman Problem and the Chinese Postman Problem. However, we are interested in a variant in which there are multiple vehicles. A particularly interesting variant is the Capacitated Arc Routing Problem (Belenguer and Benavent 1998), where each vehicle has the capacity to process a fixed number of edges, although it can traverse other edges without processing them (for example, a gritting lorry might carry sufficient grit to coat a fixed number of roads, but be able to drive along many more). Modifying the domain to model the new behaviours is straightforward: the move action is duplicated, offering a second version that can process the link it follows, subtracting one from the processing capacity of the moving vehicle. The effect is to process the link (in both directions). The goal is to have all links processed (in either direction). We start the vehicles at one location and, in one problem, also end the vehicles at the same place (a depot). Our experiments are performed on a 6x6 grid. Note that we left the constraint in place that vehicles cannot move in the opposite direction to their current facing. It seems unlikely that this makes the problem harder, but it is not a constraint usually included in CARP.

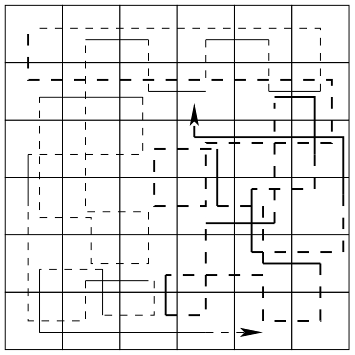

Figure 10: Routes planned for two vehicles in a 6x6 grid, visiting every edge (dashed lines indicate where vehicles are processing and solid lines where they are simply moving).

In order to harness the power of temporal reasoning in POPF we encoded the problem so that movement with processing takes longer than moving without. This encourages POPF to avoid using processing capacity when it is unnecessary to do so. Results for several problem instances are shown in Table 2. As can be seen, POPF generates good plans for cases in which the vehicles process without returning to their starting point. The addition of this goal prevents POPF from solving the problem. This appears to be linked to the fact that this additional goal makes it much harder for the relaxed plan heuristic to perceive that moving away from the starting location can help to achieve goals. Switching to a final destination in the opposite corner allows POPF to solve the problem, but with a noticeable deterioration in the plan quality compared with the cases in which no destination is specified. An example of routes for two vehicles each processing half the edges, produced by POPF is shown in Figure 10.

An example of a good dedicated solver for CARP is the Repair-Tabu Solver (RTS) (Mei, Tang, and Yao 2009), which uses a local search and tabu list approach. Reported results indicate solutions can be found to problems on graphs of similar size to the 6x6 grid in under a second.

5.1 New Encoding

In investigating the behaviour of the planners on specific problem instances we observed that the relaxed plan heuris-

Planner	POPF		LPG-td-1.0		SGPlan 5.2.1	
Problem	Makespan	Time(s)	Makespan	Time(s)	Makespan	Time(s)
V2 30,30	73	1.78	109	389.32	–	–
V2 20,40	122	4.27	140	714.09	–	–
V2 60,60	73	1.78	123	4.26	73	0.26
V4 all 60	38	19.07	62	4.01	55	1.38
V4 all 15	43	19.37	68	373.96	–	–
V2 60,60 Ret	–	–	109	46.76	–	–
V2 60,60 Opp	109	35.31	111	6.76	–	–

Table 2: Results for various CARP instances on 6x6 grids. Problem lists number of vehicles (2 or 4), capacities for each vehicle and destinations if specified (Ret = return to start, Opp = opposite corner). SGPlan could not solve any of the constrained instances within 5 minutes.

tic leads to poor behaviour in some situations, because of the separation of the vehicle facings and locations. The reason is that it can sometimes appear advantageous to achieve a facing using a local move, even though that facing will actually be irrelevant once the vehicle arrives at the location where the facing is required. By modifying the encoding to couple the facing and location into a single predicate we can improve the informedness of the heuristic. This does not appear to benefit LPG-td, but improves POPF performance in both time and quality, so that, for example, the 10x10 case can be solved in 2.9 seconds (5 times faster), with makespan of 169 (improved by 4% of optimal solution) and the 9x9 case can be solved in 1.57 seconds (better than 5 times faster) with a 17% improvement in quality.

6 Conclusions

We have presented a broad exploration of the performance of a range of temporal planners on a collection of different variants and instances of problems inspired by a client with a real need. As can be seen, even in problems with no required concurrency, failure to integrate the planning and scheduling elements of temporal planning can severely compromise the quality of the plans produced. The problems we have explored are often solved, in the literature, using local search techniques and it is therefore not entirely surprising that LPG-td performs well in solving them. However, LPG-td suffers from the disadvantages of having widely variable performance and having poor time-performance on several of the constrained variants of the problem.

Our results show that POPF demonstrates a remarkable resilience in the face of a range of different variations in the problem structure and constraints on the resources to be used to solve the problem instances. It generates good quality solutions very fast. Its success is a consequence of an informed heuristic, that gives some insight into the impact of its planning choices on the predicted duration of the plan, together with the use of a partially ordered plan construction, allowing it to postpone commitment to ordering of certain actions.

The results show both the extent to which progress in planning has supported and promoted the development of capable temporal and numeric planners, but also the sparseness of the range of planners capable of dealing effectively with this combination. Thus, we propose that this work should be seen as both an endorsement of the successes in developing informed and powerful heuristic techniques and also a challenge to continue progress in constructing new planners able to handle expressive levels of PDDL.

References

Belenguer, J. M., and Benavent, E. 1998. The capacitated arc routing problem: Valid inequalities and facets. *Comput. Optim. Appl.* 10(2):165–187.

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

Coles, A. I.; Fox, M.; Long, D.; and Smith, A. J. 2008. Planning with problems requiring temporal coordination. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 08)*.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2009. Extending the use of inference in temporal planning as forwards search. In *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 09)*.

Do, M. B., and Kambhampati, S. 2003. Sapa: A multi-objective metric temporal planner. *J. Artif. Intell. Res. (JAIR)* 20:155–194.

Edelkamp, S., and Jabbar, S. 2006. Cost-optimal external planning. In *Proceedings of In 21st National (American) Conference on Artificial Intelligence (AAAI)*. AAAI Press.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2009. Using the context-enhanced additive heuristic for temporal and numeric planning. In *Proceedings of 19th International Conference on Automated Planning and Scheduling (ICAPS 2009)*. AAAI Press.

Gerevini, A.; Saetti, A.; and Serina, I. 2006. An approach to temporal planning and scheduling in domains with predictable exogenous events. *J. of AI Research* 25:187–231.

Gerevini, A.; Saetti, A.; and Serina, I. 2010. Temporal planning with problems requiring concurrency through action graphs and local search. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS-10)*, to appear.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *J. of AI Research* 14:253–302.

Li, F.; Golden, B. L.; and Wasil, E. A. 2007. The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Computers & OR* 34(10):2918–2930.

Mei, Y.; Tang, K.; and Yao, X. 2009. A global repair operator for capacitated arc routing problem. *Trans. Sys. Man Cyber. Part B* 39(3):723–734.

Younes, H. L. S., and Simmons, R. G. 2003. VHPOP: Versatile heuristic partial order planner. *Journal of Artificial Intelligence Research (JAIR)* 20:405–430.

A Domain Encoding

```
(:durative-action move
 :parameters (?v - vehicle ?dv - direction
              ?dn - direction ?x - location
              ?y - location ?d - direction)
 :duration (= ?duration (/ 2 (speed ?v)))
 :condition (and
        (at start (>= (fuel-level ?v) (fuel-efficiency ?v)))
        (at start (vehicle-at ?v ?x))
        (at start (linked ?x ?y ?d))
        (at start (vehicle-face ?v ?dv))
        (at start (opp ?d ?dn))
    (at start (uneq ?dv ?d)))
 :effect (and
        (at start (decrease (fuel-level ?v) (fuel-efficiency ?v)))
        (at start (not (vehicle-at ?v ?x)))
        (at end (vehicle-at ?v ?y))
        (at end (vehicle-face ?v ?dn))
        (at start (not (vehicle-face ?v ?dv)))))

(:durative-action search
 :parameters (?v - vehicle ?x - location)
 :duration (= ?duration 1)
 :condition (over all (vehicle-at ?v ?x))
 :effect (at end (visited ?x))))
```