Improving Planner Performance in Grid Worlds with Macro Actions

Matthew Crosby[1] and **Ronald P. A. Petrick**[2]

Abstract. In this paper we explore a class of grid world planning domains that models high-level multi-robot navigation in confined spaces, and that gives rise to certain problem instances over which some modern planning techniques perform surprisingly poorly. We show that the inclusion of macro actions, inspired by techniques that humans use to solve similar problems, allows planners to find solutions in some cases where they would otherwise fail. We then show that the pattern exploited to create the macro actions could potentially be exploited in the majority of planning domains used in the International Planning Competition.

1 INTRODUCTION

Current satisficing planners can find plans efficiently for a large set of planning problems. However, due to their heuristic nature, and the complexity of planning in the general case, there will always be certain problem instances on which they perform badly. In this paper, we explore a class of *grid world* problems (see, e.g., [13] or the 1998 International Planning Competition) that are relevant to planning and robotics, and have relatively simple solutions, but that are not easily solved by modern planners. We show that the addition of macro actions, inspired by human problem solving methods, can significantly improve the performance of existing planners on certain instances of these problems. We also show that the patterns underlying the macro actions for the grid world problem occur in a majority of the planning problems used in the International Planning Competition.

Figure 2 presents three planning problems in which robots must navigate a grid world. If we consider how humans typically view such problems, the situations in Figure 2 (especially P1 and P2) appear to be of roughly similar difficulty. One might expect that being able to solve one of these problems might suggest that the others would be solvable in a somewhat similar manner and time frame. However, from a practical planning point of view, the latter two problems are much harder to solve: P1 is solved in less than three seconds by the planners we tested, while P2 and P3 are not solvable within an hour. One reason for this poor performance on the latter problems is that moving blindly towards the goal area is often a poor move, which conflicts with modern heuristic planning techniques that use the relaxed distance to the goal as a heuristic estimate.

One solution for P1 is to first move the robots A through E next to their goal positions; the robots e through a can then move into their goal positions, and finally A through E can be moved to their goal positions. For P2, an example solution is to move all the robots into the centre so that robots A through E are one cell to the right of their original locations, and robots a through e are one cell to the left. From this position the robots can be moved one at a time into their goal locations. P3 is a little trickier (especially for planning) as there are two possible routes round the outside which are of equal value in terms of getting towards the goal; however, in an ideal solution the robots should all go in the same direction.

```
(:action move
   :parameters (?a - agent  ?x - loc  ?y - loc)
   :precondition
      (and (at ?a ?x) (free ?y) (connected ?x ?y) )
   :effect
      (and (not (at ?a ?x)) (at ?a ?y)
           (free ?x) (not (free ?y)) )
```

Figure 1: The `move` action used in the planning problems.

2 BACKGROUND AND METHODOLOGY

In describing the example solutions above, several move actions were naturally concatenated together when they involved the same robot. This simplifies the problem (and the solution description) because we do not need to consider the places moved through by a single robot. This observation has an important consequence in terms of our encoding of the `move` action in Figure 1: all (non-static) conditions can be provided by a previous application of the action and a subsequent application can remove any intermediate positive effects.

There are many methods from the literature that are closely related to this observation. For instance, Nissim et al. [9] show how a multiagent decomposition-based method with tunnelling can be used to improve search in optimal planning. Tunnelling can be used when it can be shown that performing an action in an optimal plan can only be correct if followed up by certain other actions; those actions can then be automatically applied, refining the search space. While this is not the case for our problems, a similar idea can be used to justify combining move actions. A multiagent approach also has potential in these domains. A minor variation of [4] showed some small improvements but not as much as the method presented in this paper.

Another approach is to use an external problem solver integrated with the planner [5, 6, 10]. This approach could calculate available combinations of move actions during planning. Building macro actions through action-sequence memorisation [3] or other methods [2] also offers a potential solution. Finally, a planner like Macro-FF [1] computes macros from previous plans based on other problem instances (see also [12]) and then tries to use them to solve harder problems in the same domain.

In our work, we are interested in automatically generating macro actions for a given domain, rather than learning then from a training set of problems. Following our above observation, and as a first step, we rewrote our example domains by adding a path predicate

[1] School of Informatics, University of Edinburgh, United Kingdom, email: `m.crosby@ed.ac.uk`
[2] School of Informatics, University of Edinburgh, United Kingdom, email: `rpetrick@inf.ed.ac.uk`

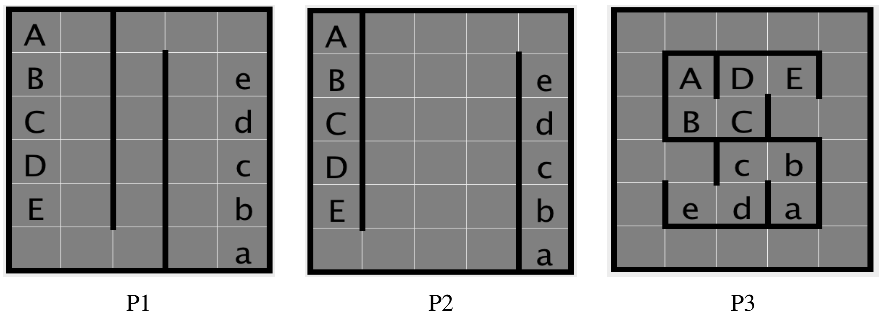

Figure 2: In this grid world problem the robots (represented by letters) must swap places with their counterparts (e.g., A with a). They can only move to adjacent grid squares (not through walls) and cannot move through each other.

which represents a route between two separate grid-squares via three intermediate steps. This path size was chosen because the standard compilation of FF allows for predicates of maximum arity 5 (i.e., one for each location in the path). Using paths in this way means that a `move4` action can be included where a robot can move four squares at once (provided all intermediate spaces are empty and not blocked by walls). For example, we may have `(path x1y1 x2y1 x2y2 x3y2 x3y3)` that can be used to move a robot from `x1y1` to `x3y3` as long as all the intermediate places are free. A `move8` action was also included which combines two `move4` actions.

A simple algorithm was written to pre-calculate all valid paths in the domain (respecting walls) that did not repeat locations. The original `move` action was left in the domain so that the robots could also be moved just one space if required. This means that the macro domain contained the actions `move`, `move4`, and `move8` only. Extending this to a generalised `moveN` action based on an external path calculator is left for future work.

3 EVALUATION

The results of our approach are presented in Table 1. In this initial study, Macro-FF [8, 1] (m-FF in the table) was used because it contains optimisations for use with similarly constructed macro actions. Fast Downwards' implementation of FF and LAMA [7] was also used for comparison, in order to include a recent and strongly performing planner with a different heuristic. The first three parts of Table 1 correspond to problems P1-P3, while the last part presents results on the visit-all domain from IPC11. The problems varied in the number of robots in the domain. For P1 and P2, the number of rows was varied so that it always was equal to the number of agents plus one while the overall structure remained the same. For P3, only the number of agents in the domain was changed.

From the results for P1 we can see that adding macro actions causes some overhead for the planners, especially for m-FF. This is because of the extra path information which is especially problematic in the more open domains. The results for P2 show how macro actions help solve previously unsolvable problems. The coverage on the right hand side of the table is much greater than that on the left hand side. Finally, the macro actions also help find solutions for P3 but this is still a much harder problem and unsolvable with ten robots.

In visit-all, the problem is to visit all the places on a grid world. Our approach somewhat trivialises the visit-all domain because there is only one robot so the intermediate locations do not need to be stored to be checked. From the results we can see that this allows the FF planners to easily solve problems that were previously too large.

By analysing the `move` action, we are also investigating the following properties that could potentially allow for the automatic generation of macro actions: 1) all preconditions of a latter action are either added by the previous action or compatible with the preconditions of the first action in a sequence, and 2) all positive effects of the first action are either removed by the end of the subsequent action or do not appear in any subsequent action. We also checked the much stronger property, where the first condition of each clause is satisfied, and found that 62 of the 81 benchmark domains contain action pairs that satisfy this condition. This lends evidence to our belief that this approach is potentially applicable in many domains, though there is still much work to be done to verify this claim. We believe that progress can be made by analysing unground operators while taking into account knowledge of the static predicates in the domain.

Table 1: Time (in seconds) for running multiple planners over the standard and macro versions of the problems. A 'dash' means that the planner did not return a result within 10 minutes.

	Standard			Macro Actions		
	lama	fd-FF	m-FF	lama	fd-FF	m-FF
P1-3	0.03	0.01	0.63	0.03	0.02	0.28
P1-4	0.01	0.02	3.54	0.48	0.37	3.05
P1-5	0.16	0.07	2.37	0.47	0.85	64.78
P1-6	1.13	0.23	26.19	2.94	10.63	–
P2-3	2.40	0.53	0.06	9.10	0.36	0.10
P2-4	–	–	11.03	215.65	4.58	0.27
P2-5	–	–	–	–	19.83	7.46
P2-6	–	–	–	–	366.36	32.16
P3-3	3.2	1.54	77.03	14.17	7.13	29.14
P3-4	–	–	–	46.04	370.27	35.59
P3-5	–	–	–	–	–	–
P3-6	–	–	–	–	–	–
V16	0.18	16.19	25.12	0.51	0.43	0.31
V20	0.58	–	–	2.02	1.69	1.04
V24	1.05	–	–	6.08	4.72	3.22

4 CONCLUSION AND FUTURE WORK

We have shown that by directly encoding macro actions in certain types of grid world domains, planning problems can go from unsolvable to solvable in a few seconds. Moreover, these macro actions can potentially be constructed whenever the concatenation of multiple actions leads to a simplified overall result (i.e., some added effects are then removed or vice versa). We intend to explore this space further in future work, implementing an automated algorithm for generating macro actions that incorporates ideas from some of the other approaches in the literature (e.g. [11]), and testing this approach over a larger class of domains and related planners (e.g., Marvin [3]).

Acknowledgements: The research leading to these results has received funding from the European Union's Seventh Framework Programme under grant agreement no. 610917 (STAMINA).

REFERENCES

[1] Adi Botea, Markus Enzenberger, Martin Müller, and Jonathan Schaeffer, 'Macro-FF: Improving AI Planning with Automatically Learned Macro-Operators', *Journal of Artificial Intelligence Research*, **24**, 581–621, (2005).

[2] Lukáš Chrpa, Mauro Vallati, and Thomas Leo McCluskey, 'MUM: A Technique for Maximising the Utility of Macro-operators by Constrained Generation and Use', in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 65–73, (2014).

[3] Andrew Coles and Amanda Smith, 'Marvin: A Heuristic Search Planner with Online Macro-Action Learning', *Journal of Artificial Intelligence Research*, **28**, 119–156, (2007).

[4] Matt Crosby, Michael Rovatsos, and Ronald P. A. Petrick, 'Automated Agent Decomposition for Classical Planning', in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 46–54, (2013).

[5] Christian Dornhege, Patrick Eyerich, Thomas Keller, Sebastian Trüg, Michael Brenner, and Bernhard Nebel, 'Semantic Attachments for Domain-Independent Planning Systems', in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 114–121, (2009).

[6] Esra Erdem, Kadir Haspalamutgil, Can Palaz, Volkan Patoglu, and Tansel Uras, 'Combining High-Level Causal Reasoning with Low-Level Geometric Reasoning and Motion Planning for Robotic Manipulation', in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4575–4581, (2011).

[7] Malte Helmert, 'The Fast Downward Planning System', *Journal of Artificial Intelligence Research*, **26**, 191–246, (2006).

[8] Jörg Hoffmann and Bernhard Nebel, 'The FF Planning System: Fast Plan Generation Through Heuristic Search', *Journal of Artificial Intelligence Research*, **14**, 253–302, (2001).

[9] Raz Nissim, Udi Apsel, and Ronen Brafman, 'Tunneling and Decomposition-Based State Reduction for Optimal Planning', in *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pp. 624–629, (2012).

[10] Ronald P. A. Petrick and Andre Gaschler, 'Extending Knowledge-Level Contingent Planning for Robot Task Planning', in *ICAPS 2014 Workshop on Planning and Robotics (PlanRob)*, (2014).

[11] Bram Ridder and Maria Fox, 'Heuristic Evaluation Based on Lifted Relaxed Planning Graphs', in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 244–252, (2014).

[12] Earl D Sacerdoti, 'Planning in a hierarchy of abstraction spaces', *International Joint Conference on Artificial Intelligence*, **3**, 412–422, (1973).

[13] Craig Tovey and Sven Koenig, 'Gridworlds as testbeds for planning with incomplete information', in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 819–824, (2000).