Improving Planning Performance in PDDL+ Domains via Automated Predicate Reformulation

Santiago Franco, Mauro Vallati[0000−0002−8429−3570], Alan Lindsay, and Thomas L. McCluskey

School of Computing and Engineering
University of Huddersfield, Huddersfield, United Kingdom
{n.surname}@hud.ac.uk

Abstract. In the last decade, planning with domains modelled in the hybrid PDDL+ formalism has been gaining significant research interest. A number of approaches have been proposed that can handle PDDL+, and their exploitation fostered the use of planning in complex scenarios. In this paper we introduce a PDDL+ reformulation method that reduces the size of the grounded problem, by reducing the arity of *sparse* predicates, i.e. predicates with a very large number of possible groundings, out of which very few are actually exploited in the planning problems. We include an empirical evaluation which demonstrates that these methods can substantially improve performance of domain-independent planners on PDDL+ domains.

Keywords: Automated Planning · Hybrid Reasoning · Reformulation

1 Introduction

Automated planning is one of the most prominent Artificial Intelligence (AI) challenges; it has been studied extensively for several decades and has led to a large number of real-world applications. The growing number of domain-independent PDDL+ planners is fostering the exploitation of planning in complex real-world applications, where notions of continuous processes and discrete events and actions are needed [4]. The use of reformulation and configuration techniques, which can automatically re-represent the planning model in order to increase efficiency and enable a scale up in size of applications that can be handled. In the last decades, research into reformulation techniques has attracted significant attention. Types of reformulation of classical PDDL models include macro-learning [6] and configuration [9].

Hybrid PDDL+ models are amongst the most advanced models of systems and the resulting problems are notoriously difficult for planners to cope with due to non-linear behaviours and immense search spaces. Complexity is exacerbated by the potentially huge size of the fully grounded problems, needed by planners in order to effectively explore the search space, which can make some

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_42

Algorithm 1 Reformulation for flattening sparse predicates

Require: D_o, I_o, s^t, a^t
Ensure: D_r,I_r
1: $SP = statics(D_o); P = predicates(D_o) \cup functions(D_o)$
2: $D_r = D_o; I_r = I_o$
3: **for all** p_j **in** P, **where** $arity(p_j) > 2$ **do**
4: **if** $sparsity(p_j, I_o) > s^t$ **then**
5: $p_{stat} = findConstrainingStatic(p_j, SP)$
6: **if** $p_{stat} \neq None$ **then**
7: $T_{p_{stat}} = getSparseVariables(p_{stat}, I_o, a^t)$
8: $C^{new} = makeConstants(T_{p_{stat}}, s_o)$
9: $D_r = addAsConstants(D_r, C^{new})$
10: $D_r = updateOpProEv(D_r, T_{p_{stat}}, C^{new})$
11: $I_r = updatePredsFuncs(D_r, I_r, T_{p_{stat}}, C^{new})$

problems impossible to tackle. Particularly, grounding is also strongly affected by predicates' instances that will not be reachable in any state of the problem.

In this paper we introduce a PDDL+ reformulation method that allows to drastically reduce the size of the grounded problem, by reducing the arity of *sparse* predicates, i.e. predicates with a very large number of possible groundings, out of which very few are actually exploited in the planning problems. Arity is reduced by merging suitable objects together, and partially grounding the operators, processes and events in which reformulated predicates are involved. Our experimental analysis, performed on a range of problem from different application domains, shows that the proposed reformulation technique can substantially improve the performance of PDDL+ planning engines, by allowing problems to be grounded and by constraining the search space.

2 The Proposed Reformulation Approach

Our approach relies on identifying sparse predicates that are partially constrained by a static predicate. Through combining the sparsity measure for dynamic predicates with a constraining static predicate, the approach is able to better identify predicates for which the reformulation will have significant impact.

Let us consider an hybrid version of the well-known Rovers domain model, where movements and energy generation via solar are modelled as continuous processes, triggered by actions under the control of a planner, and constrained by appropriate events. In the Rovers domain, rovers are used to make soil and rock samples and to take pictures for various objectives. This requires that the rovers are moved between waypoints in order to establish shots and collect samples from certain positions. The constraints establishing the properties of the rovers and the relationships between waypoints (e.g., that a rover can traverse between waypoints) are encoded as static facts. As with many network based relationships, only a fraction of the potential connections are made available in any

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_42

particular problem model. Of course, as the number of waypoints (and rovers) grows this fraction will reduce. The problem model reformulation reported in this paper collapses the variables of predicates, creating a model with fewer sparsely instantiated predicates. This procedure can be applied to Rovers, for example, consider replacing the (arity 2) predicate: (`visible` *?waypoint1 ?waypoint2*) with an alternative (arity 1) encoding: (`visible` *?visible_waypoint1_waypoint2*). Whereas in the original version, the domain of possible instantiations is every pair of waypoints; in the second approach, we can reduce the domain by only defining constants for the combinations of waypoints for which the relation holds.

2.1 The Reformulation Algorithm

Algorithm 1 shows how the reformulation of a domain model, D_o, and a problem model, I_o, is performed. Beside the models to be reformulated, the algorithm requires as input a sparsity threshold s^t, which is used to decide whether or not it is useful to perform the reformulation and a parameter, and a^t, which sets the maximum number of variables considered in the reformulation (how these parameters are set is explained below).

In the algorithm (see Algorithm 1) the sparsity of the predicates (Boolean or numeric) with arity greater than 2 are assessed in turn (line 3) to determine if they are suitable for the reformulation step. As a measure of sparsity we compare the set of propositions in the initial state with the possible set of all propositions for the predicate. For example, if we consider a specific example Rovers problem from our benchmark problems, with 4 waypoints and 2 rovers, we can calculate the total set of possible propositions as: $4 \times 4 \times 2 = 32$. In this example, there are 10 instances of `can_traverse` in the initial state and so the sparsity for this predicate is 10/32.

In the case of a sparse predicate, p_j, the procedure attempts (line 5) to find a static predicate, p_{stat}, such that p_j is only used in transition schemas (that is in the action, process or event schemas) with p_{stat}. We consider predicates as static if instances of the predicate can not be deleted or created during the planning process but, in the case of numeric predicates, their value can be changed. If there is more than one constraining static predicate then one is selected heuristically by selecting the predicate that occurs the most in transition schemas. There are two static facts that constrain the `can_traverse` predicate: `can_traverse` itself and `visible`. The algorithm selects `visible` as it appears in more transition schemas.

2.2 Reformulating the domain and problem models

In the case that p_{stat} exists (e.g., `visible`), a reformulation step is applied using p_{stat} as the basis. In our current system, we have considered subsets of the variables of the static facts and so we add a parameter, a^t, to determine the maximum arity of the reformulation. The best $\texttt{max}(a^t)$ variables are selected (line 7) using the sparsity of the tuple for p_{stat} in Io. We use T_p to denote a subset of the variables of p. In our example, `visible` has arity 2 and therefore $T_{\texttt{visible}}$

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_42

would contain both its variables. The variables in T_{stat} are then combined to form a set of constants, C^{new}, of type, t^{new}, which are added to the domain model. One constant is defined for each distinct combination of these variables for the instances of the predicate in I_o. For example, a new constant is generated for each distinct combination of the waypoints in the instances of the `visible` predicate in the initial state. For instance, `(visible waypoint3 waypoint0)` leads to a new constant `waypoint3_waypoint0` (using the new type).

At this stage (line 10) each of the transition schemas that refer to p_{stat} are reformulated. For example, in Rovers, the transitions with `visible` as a precondition are identified (e.g., `start-navigate`, `communicate_soil_data`). For each predicate (dynamic, static or numeric) in these transitions the algorithm tests to determine if it can be part of this reformulation step. If the predicate is only used in transition schemas with p_{stat}, and $T_{p_{stat}}$ is a subset of the parameters of the predicate then it is selected for reformulation. In the case of `visible`, only the `can_traverse` predicate is constrained by the `visible` predicate and so only these two are selected for reformulation. For each selected predicate, p, (including p_{stat}) a new predicate, p', is made by replacing the variables that are in $T_{p_{stat}}$ with a single variable of type, t^{new} and retaining the other variables (e.g., $arity(p') = arity(p) - |T_{p_{stat}}| + 1$). For example, (`can_traverse` *?rover ?waypoint1 ?waypoint2*), is reformulated as (`can_traverse` *?rover ?new-type*). The original predicate, p, is omitted from the new model, Dr.

Each transition schema that depends on p_{stat} is partially grounded so that the variables corresponding to those in $T_{p_{stat}}$ are grounded and constants added as necessary (i.e., for referencing the individual objects). This allows the relation between the new constants and the original objects to be maintained. For example, there are new `start-navigate` operators for each of the new constants, e.g., `start-navigate-waypoint3-waypoint2`. In `start-navigate`, each matching of *?waypoint* is added as constant as it is referred to in other predicates.

Finally, the problem model is reformulated (line 11) by changing those predicates involved in the reformulation to use the constants in C^{new} in the initial state and goal, using a similar approach as described above. Of course, after this step has been applied once, the procedure can be repeated on the reformulated model supporting further combining of variables as appropriate.

3 Experimental Analysis

Four PDDL+ planners at the state of the art are included in the evaluation: ENHSP [8] , UPMurphi [3], DINO [7], and SMTPlan [2].

All reported results were achieved by running the planners on a machine equipped with i7-4750HQ CPU, 16 GB of memory, running Ubuntu 16.10 OS. 4 GB of memory were made available for each run, and a 15 CPU-time minutes cut-off time limit was enforced.

The experimental evaluation is performed by considering three benchmark domains, namely Hybrid Rover, Urban Traffic Control, and Baxter.

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_42

Table 1. CPU-time seconds needed by the planners to find a satisficing solution. O (R) rows show the results achieved when running the Original (Reformulated) model. X indicates grounded but not solved. "–" means crashed during grounding. NA indicates that the planner is unable to handle the model.

Planner		**Baxter**					**Hybrid Rover**					**UTC**				
		1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
ENHSP	O	0.40	X	26.8	13.5	335.7	1.3	18.9	58.54	77.33	–	5.3	10.8	–	–	–
	R	0.45	151.7	23.5	15.2	17.9	1.1	35.0	60.28	65.5	87.5	2.7	3.2	12.5	6.9	30.8
UPMurphi	O	X	X	X	X	X	X	X	X	X	X	7.26	X	–	148.78	X
	R	X	X	X	X	X	X	X	X	X	X	0.62	5.02	29.34	5.2	49.4
DINO	O	12.0	X	X	X	X	116.24	X	X	X	X	X	X	–	X	X
	R	27.6	X	X	X	X	X	X	X	X	X	X	X	X	X	X
SMTPlan	O	0.01	X	X	X	X	0.5	1.8	X	X	X	NA	NA	NA	NA	NA
	R	0.01	X	X	X	X	0.5	1.8	X	X	X	NA	NA	NA	NA	NA

The **Baxter** domain [1] exploits planning for dealing with articulated objects manipulation tasks. The "simplified" domain model has been extended by allowing continuous movements of a joint, modelled via actions and process envelope, on different axis, and by adding events for preventing movements wider than 360 degrees. Problems consider articulated objects composed by 2–5 links. The objects of type `link` has been merged into a new type, and four predicates have then been reformulated: `connected`, `increasing_angle`, `decreasing_angle`, and `use`. Our reformulation approach has been applied following the fact that the `connected` predicate is static, and is exploited in operators and processes to control all the other mentioned predicates. According to the results shown in Table 1 UPMurphi, DINO, and SMTPlan grounded and explored the search space of all the considered problems but only ENHSP solved most of the problems using the original representation. ENHSP solved all but one of the problems using the original models. Remarkably, the use of reformulated models did lead to a significant search speedup, and allows ENHSP to solve all the considered benchmarks. Empirical evidence indicates that the reformulation allows to improve the pruning done by the reachability analysis of ENHSP, leading to a faster expansion and evaluation of the search states. The other planners could only solve the simplest problem in both original and reformulated versions. Interestingly, DINO works better with the original representation. According to our observations, in that specific problem, the DINO heuristic expanded twice the number of states compared to its use with the reformulated model, but to find the same solution. This seems to suggest that, on some simple problem instances, the use of reformulated models can reduce the effectiveness of a domain-independent heuristic.

We extended the well-known **Rover** domain model introduced in IPC-3 by modelling as continuous processes the movements of the rovers, and the energy generation via solar power. Each of the mentioned processes can be controlled by the planner using two actions, and is constrained, where appropriate, via events. The predicate `can_traverse` has been reformulated by merging the objects of type `waypoints`, as shown in the previous section. The use of reformulated models allows ENHSP to solve a larger number of problem instances. However, in few

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_42

Table 2. Ratio of maximum search space sizes of original vs reformulated representations for the UPMurphi and DINO planners. "–" is used to indicate cases where one of the approaches lead planners to crash during grounding.

	Baxter					**Hybrid Rover**					**UTC**				
Problem	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Ratio	2.42	3.98	3.98	3.13	4.76	1.00	1.05	1.88	2.86	4.18	18.59	37.08	–	18.58	21.05

cases, the reformulation negatively affected search performance, once the grounding is completed. The larger the problem, the closer the performance between both versions got. According to our observations, the default heuristic exploited by ENHSP is less informative, for this domain, when exploited on reformulated models. However, one large and complex problems, the gap is closed by the fact that the reformulated model allows the planner to explore a significantly larger size of the search space. DINO, running on reformulated models, is able to complete the grounding of all the considered problems, but lacks of the capability of generating plans. SMTPlan is not significantly affected by the different domain models. This seems to be related to the compilation of the PDDL+ model into an SMT encoding that allows to reduce grounding-related issues.

Finally, the **Urban Traffic Control** (UTC) domain [5] models the use of planning for generating traffic light signal plans, in order to de-congest a urban region. In this analysis we considered the problems introduced by McCluskey and Vallati [5], which involved a network of 10 junctions, and we extended it by considering problems with 20 and 30 junctions, obtained by connecting identical regions. Problems 1–3 have 10, 20 and 30 junctions respectively and only one goal. Problems 4 and 5 have 10 and 20 junctions respectively, both of them have 3 goals. Goals in this domain indicates the requirement of reducing the congestion on a specific link of the network. In this domain, the predicate `flowrate` has been reformulated by merging the objects of type `link` into a new type, which represents road links which are connected via a junction. ENHSP and UPMurphi were run using the heuristic proposed by McCluskey and Vallati [5]. Results presented in Table 1 indicate that reformulation has a strong beneficial impact on planners' performance. ENHSP and UPMurphi are able to quickly solve problems involving large networks as they can manage to ground the problem. In ENHSP most of the improvement is due to the faster grounding, and on the largest 30 junction problem 3, to be able to ground it at all. On the contrary, in the case of UPMurphi and DINO, the reformulation boosts also the search performance, as also the size of each state is significantly reduced by the reformulation. Unsurprisingly, the domain-independent heuristic exploited by DINO is not very informative in UTC problems, but the planner is able to ground and to explore a large area of the search space when run on reformulated models. SMTPlan is not able to handle the UTC domain model.

Impact of Reformulation on Search Space Size. Beside the impact on planning performance, it is also important to assess whether the use of the proposed reformulation approach allows to actually reduce the size of a search

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_42

state, so that the search space can be explored quicker and more efficiently. Table 2 shows how planners DINO and UPMurphi benefit from the reformulation of PDDL+ models, in terms of state size. We consider these planners because they allow to measure effectively and accurately the size of a single search state. Results are presented in terms of ratio of maximum space sizes between original and the corresponding reformulated representation; for instance, a value of 1.0 indicates that there is no difference, while a value of 2.0 means reformulated search can create twice the number of states before running out of memory. In almost every considered instance, reformulation greatly increases the maximum state space, and therefore allows the planners to explore a larger portion of the space, and to generate search states in less time.

4 Conclusion

In this paper, we introduced a reformulation approach that allows to reduce the size of a PDDL+ grounded problem by tackling the arity of sparse predicates. Our experimental analysis showed that the proposed reformulation: (i) effectively reduces the grounding size of hybrid problems, hence allowing planners to deal with them; and (ii) positively affect the size of each search state, leading to a faster and more effective exploration of the search space. Results suggest that PDDL+ reformulation techniques, by allowing larger problems to be reasoned upon by planning engines, can foster the exploitation of planning in real-world applications. Future work is planned on extending the number of objects that can be merged, and to study the importance of the sparsity threshold.

References

1. Capitanelli, A., Maratea, M., Mastrogiovanni, F., Vallati, M.: On the manipulation of articulated objects in humanrobot cooperation scenarios. Robotics and Autonomous Systems **109**, 139 – 155 (2018)
2. Cashmore, M., Fox, M., Long, D., Magazzeni, D.: A compilation of the full PDDL+ language into SMT. In: Proc. of ICAPS (2016)
3. Della Penna, G., Magazzeni, D., Mercorio, F., Intrigila, B.: UPMurphi: A tool for universal planning on PDDL+ problems. In: Proc. of ICAPS (2009)
4. Fox, M., Long, D.: Modelling mixed discrete-continuous domains for planning. Journal of Artificial Intelligence Research **27**, 235–297 (2006)
5. McCluskey, T.L., Vallati, M.: Embedding automated planning within urban traffic management operations. In: Proc. of ICAPS (2017)
6. Newton, M.A.H., Levine, J., Fox, M., Long, D.: Learning macro-actions for arbitrary planners and domains. In: Proc. of ICAPS (2007)
7. Piotrowski, W.M., Fox, M., Long, D., Magazzeni, D., Mercorio, F.: Heuristic planning for PDDL+ domains. In: Proc. of IJCAI (2016)
8. Scala, E., Haslum, P., Thiébaux, S., Ramírez, M.: Interval-based relaxation for general numeric planning. In: Proc. of ECAI (2016)
9. Vallati, M., Hutter, F., Chrpa, L., McCluskey, T.L.: On the effective configuration of planning domain models. In: Proc. of IJCAI (2015)

ICCS Camera Ready Version 2019
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-22750-0_42