Automated Planning Encodings for the Manipulation of Articulated Objects in 3D with Gravity

Riccardo Bertolucci[1], Alessio Capitanelli[2], Marco Maratea[2], Fulvio Mastrogiovanni[2], and Mauro Vallati[3]

[1] DeMaCS, University of Calabria, Italy
bertolucci@mat.unical.it
[2] DIBRIS, University of Genova, Genova, Italy
{name.surname}@unige.it
[3] University of Huddersfield, UK
m.vallati@hud.ac.uk

Abstract. The manipulation of articulated objects plays an important role in real-world robot tasks, both in home and industrial environments. A lot of attention has been devoted to the development of *ad hoc* approaches and algorithms for generating the sequence of movements the robot has to perform in order to manipulate the object. Such approaches can hardly generalise on different settings, and are usually focused on 2D manipulations.
In this paper we introduce a set of PDDL+ formulations for performing automated manipulation of articulated objects in a three-dimensional workspace by a dual-arm robot. Presented formulations differ in terms of how gravity is modelled, considering different trade-offs between modelling accuracy and planning performance, and between human-readability and parsability by planners. Our experimental analysis compares the formulations on a range of domain-independent planners, that aim at generating plans for allowing a dual-arm robot to manipulate articulated objects of different sizes. Validation is performed in simulation on a Baxter robot.

1 Introduction

The manipulation of articulated objects plays an important role in real-world robot tasks, both in home and industrial environments [18,15]. In literature, the problem of determining the two- or three-dimensional (2D or 3D) configuration of articulated or flexible objects has received much attention in the past few years [28,21,6,7], whereas the problem of obtaining a target configuration via manipulation has been explored in motion planning [30,3,26]. However, the employed manipulation strategies are often crafted specifically for the problem at hand, with the relevant characteristics of the object and robot capabilities being either hard coded or assumed, thus undermining generalisation and scalability. More general solutions [1,7] are limited to 2D configuration, with a partial exception for the work in [1], where the notion of *overlap* between different parts of a cable is explicitly considered. A challenging aspect of identifying the movements needed to achieve a desired 3D configuration of an articulated object manipulation is that the effect of gravity has to be explicitly modelled and taken into account.

In this paper we introduce a set of PDDL+ [12] models for performing an automated manipulation of articulated objects in a 3D workspace by a dual-arm robot. Presented models differ in terms of how gravity is modelled, considering different levels of accuracy, and in the design of the formulation itself, by trading-off between human-readability and usability by planners. Three different levels of complexity are designed and described for modelling the impact of gravity on the articulated object.

Our experimental analysis compares the proposed models on a range of domain-independent PDDL+ planners, taking into account articulated objects with different sizes and different variable parameters (e.g., acceleration value) for each level of complexity. As a matter of fact, the empirical analysis highlights that the proposed models, designed following the more natural and concise representation of the considered problem, can not be handled by most of the selected PDDL+ planning engines. For this reason, we then modify the introduced PDDL+ models, providing a formulation which trades the intuitiveness of the model for the acceptability by planning engines.

Results of our extensive experimental analysis show to which degree the proposed PDDL+ models allow domain-independent PDDL+ planning engines to solve tasks that model practical, real-world applications, i.e., in terms of robot workspace and the number of links and physical features characterising the articulated object. Moreover, it gives also an indication about what is the most suited formulation and planner to solve these specific problem instances.
To sum up, the main contributions of this paper are:

- We define two sets of PDDL+ models for the task of automated, robot-based manipulation of articulated objects in a 3D workspace. Each set of models considers three different levels of complexity for representing the effect of gravity.
- We analyse the performance of a number of domain-independent PDDL+ planners on realistic articulated object manipulation tasks.

We also validate generated plans using a robot control architecture for a dual-arm robot manipulator in simulation. As a side effect of our work, we provide a challenging domain, and its PDDL+ models, to the planning community.

2 Problem Statement

Among the tasks typically carried out in shop-floor environments, the manipulation of flexible objects, e.g., cables [14,24], is particularly challenging. On the one hand, it is beneficial to plan the target cable configuration in advance; on the other hand, it is often necessary to keep a cable firmly using one grasping point to be able to manipulate other parts. A robot capable of manipulating flexible objects in its 3D workspace must be able to: (i) represent object configurations adopting suitable modelling assumptions, and then segment the whole manipulation problem in simpler actions to be sequenced and performed, each action operating in-between two intermediate 3D object configurations; and (ii) represent the actions to carry out using a formalism which allows for robust plan execution and modelling inaccuracies.

These requirements lead to a robot perception and control architecture characterized by the following features: (a) similarly to the approach described in [1], the robot

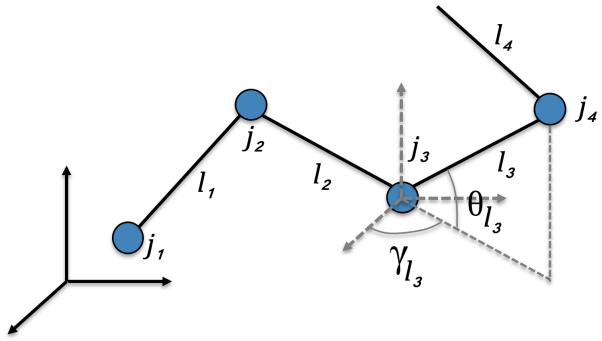

Fig. 1: A 3D articulated object configuration.

plans an appropriate sequence of actions to determine relevant 3D intermediate configurations for articulated objects (i.e., a suitable *simplified* model for a flexible object like a cable) in order to determine a target 3D configuration; and (b) during plan execution, the robot monitors the outcome of each action, and compares it with the intermediate target configuration to achieve. The problem we consider in this paper can be defined as follows: given a target object configuration in 3D space, determining a plan $\mathcal{P}$ to obtain it as an ordered set of actions $\mathcal{P} = \{a_1, \ldots, a_i, \ldots, a_N; \prec\}$, where each action a_i involves one or more 3D manipulation operations to be executed by a dual-arm robot. We pose a number of assumptions described as follows:

1. flexible objects are modelled as articulated objects with a given number of links and joints, as it is customary for computational reasons [30]; we assume an inertial behaviour, i.e., rotating one link causes the movement of all upstream and downstream links, depending on the rotation joint;
2. the effects of gravity on all articulated object's 3D configurations are explicitly considered;
3. we do not assume any specific grasping or manipulation strategy to obtain a target 3D object configuration starting from another configuration;
4. the perception of articulated objects, although affected by noise, is considered *perfect*, i.e., data association is given.

We define an articulated object as a 2-ple $\alpha = \langle \mathcal{L}, \mathcal{J} \rangle$, where $\mathcal{L}$ is the ordered set of its L links, i.e.,

$$\mathcal{L} = \{l_1, \ldots, l_j, \ldots, l_L; \prec\}, \tag{1}$$

and $\mathcal{J}$ is the ordered set of its $J = L - 1$ joints, i.e.,

$$\mathcal{J} = \{j_1, \ldots, j_k, \ldots, j_J; \prec\}. \tag{2}$$

Each link l is characterised by three parameters, namely a length, and two orientations θ_l and γ_l, expressed with respect to a robot-centred reference frame (Figure 1). We allow only for a limited number of discrete orientation values, i.e., θ_l and γ_l can take values from a pre-determined set of possible values. Given a link l_j, upstream links are

those from l_1 to l_{j-1}, whereas downstream links range from l_{j+1} to l_L. Such absolute representation leads to the direct perception of links and their orientations. When a sequence of manipulation actions is planned, changing one absolute orientation requires, in principle, the propagation of such change upstream or downstream the object via joint connections. Given an articulated object α, its configuration is a L-ple:

$$\mathcal{C}_\alpha = \{(\theta, \gamma)_1, \ldots, (\theta, \gamma)_l, \ldots, (\theta, \gamma)_L\}, \tag{3}$$

where it is intended that the orientations θ and γ are expressed with respect to an *absolute*, robot-centred, reference frame.

3 Formulation

In order to address the problem introduced above, we exploited PDDL+ to formulate three different domain models, corresponding to three different levels of abstraction of the impact of gravity on the articulated object.

PDDL+ [12] is an extension of the standard planning domain modelling language, PDDL, to model mixed discrete-continuous domains. In addition to instantaneous and durative actions, PDDL+ introduces *continuous processes* and *exogenous events*, that are triggered by changes in the environment. Processes are used to model continuous change, and therefore are well suited in this context to model the impact of gravity on articulated objects.

The absolute representation of angles we employ, on the one hand reduces the burden on the robotic framework, because the orientations of links is directly observable by the robot perception system and does not require any additional calculation. On the other hand, the complexity of the planning process is increased, due to the fact that any manipulation action has to be propagated to all the upstream or downstream link orientations. The interested reader is referred to [7] for an extensive comparison of different joint angles representation techniques in 2D setting.

In the proposed PDDL+ models, a `connected` predicate is used to describe the fact that two links are jointed. Joints are not explicitly modelled: the `connected` predicate indicates the presence of a joint between the two involved links, and the orientation is given via the `angle` function, which indicates the absolute orientation of the link l_i with regards to a plane j. The value of angles ranges between 0 and 359 degrees. The effect of the manipulation of two `connected` links are propagated via a corresponding `affects` predicate. In order to reduce the computational complexity, we fixed the way in which the robot can manipulate two connected links, so that propagation can only happen upstream. In other words, given two links, we allow the robot to move only the upstream link, while the other one is kept fixed. It should be noted that, if needed, the model can be easily extended to deal with both up and downstream manipulation by adding the appropriate predicates. The number of planes that can be represented is not fixed and can be easily modified: in our evaluation we considered 2 planes, vertical and horizontal, corresponding to a 3D space.

The planner can modify the orientation of links using the following constructs:

- An operator `start-increase(l1,l2,plane)` is used by the planner to manipulate the orientation of the link l_2 on the plane $plane$, by using a gripper for keeping l_1 still, and another gripper for moving l_2.
- A process `move-increase(l2, plane)` is used for modelling the continuous movement performed by the robot to increase the absolute angle related to l_2 on the corresponding $plane$. This process is activated by the above operator.
- An operator `stop-increase(l1,l2,plane)` is activated by the planner to stop the modification of the orientation of the l_1 and l_2 links. The robot is therefore releasing the two links.
- The events `back-to-zero(l, plane)` and `back-to-360(l, plane)` are triggered when the value of the angle of link l on $plane$ reaches, respectively, 360 or 0. In the former (latter) case, the value of the angle is reset to 0 (359).
- A process `propagate-increase(l1,l2,plane)` is activated when a process `move-increase(l2, plane)` is ongoing, and it allows to propagate the effects of the current manipulation on all the affected upstream angles.

In a nutshell, the planning engine can modify the angle between two connected links via the operator `start-increase(l1,l2,plane)`: this starts the movement process, that can be stopped by the engine using another dedicated operator. The movement process is also impacting a (potentially long) cascade of processes that models the propagation of the manipulation to affected upstream angles.

The above-listed constructs are in charge of performing and modelling manipulations aimed at increasing angles. A corresponding set of constructs is used to allow the planner to decrease some specified angles. Figure 2 shows the PDDL+ encoding of the `start-increase` operator and the `back-to-zero` event. Notably, the predicate `in-use` is exploited to avoid parallel manipulations of the articulated object by the robot. This is because the robot's grippers are not explicitly modelled, therefore many different actions could potentially be planned in parallel by the planning engine. The `freeToMove` predicates are used to indicate if a link is currently being manipulated or not; these predicates are a sort of token for grasping a specific link.

3.1 Modelling Gravity

Gravity is one of the main reasons for encoding a model in PDDL+, as gravity effects are (i) continuous in nature, and (ii) not under the direct control of the planning engine. For these reasons, PDDL+ constructs such as continuous processes and events are extremely handy for describing the impact of gravity on the 3D manipulation of an articulated object.

The representation of the effects of gravity on an articulated object can be cumbersome, and may prevent the generation of valid plans in a reasonable amount of time. Because of that, we introduce three different levels of complexity that can be implemented in the proposed PDDL+ model. It is worth noting that the typical articulated object, in order to support the manipulation via a robot, has quite stiff joints, which are therefore resisting –up to some degrees– to the gravity effect.

No Gravity (NoG). The most trivial way to reduce the complexity burden due to the computation of the effects of gravity on the articulated object is, of course, to completely ignore gravity. In cases where the joints of the articulated objects are extremely

```
(:action start-increase
:parameters (?l1 -link ?l2 -link ?x -plane)
:precondition (and  (connected ?l1 ?l2)
    (not (in-use)))
:effect (and  (in-use)
    (not (freeToMove ?l2))
    (not (freeToMove ?l1))
    (increasing_angle-robot ?l2 ?x)))

(:process move-increase
:parameters (?l2 -link ?x -plane)
:precondition
    (increasing_angle-robot ?l2 ?x)
:effect
    (increase (angle ?l2 ?x)(* #t (speed-i))))

(:event back-to-zero
:parameters (?l3 -link ?x -plane)
:precondition
    (>= (angle ?l3 ?x) 360)
:effect
    (assign (angle ?l3 ?x) 0))
```

Fig. 2: Part of the proposed PDDL+ formulation.

stiff, this model can still give some useful information to the robot. Notably, the reduced complexity may allow to quickly re-plan in cases where the robot observes that gravity has significantly modified the configuration of the object.

Uniform Circular Motion (MCU). A more sophisticated way of modelling the impact of gravity on an articulated object can be obtained by taking a joint-by-joint perspective. As links are connected by joints, they can not fall to the ground, but are bounded to each other by the joints. The impact of gravity on a joint angle can be modelled as a uniform circular motion that moves the angle towards a value of 360 (if we consider a 180-degree angle to be on the z axis). In this encoding, the angular speed is constant. The impact of gravity on an angle is modelled using a pair of dedicated processes (according to the fact that the initial angle is lower or higher than 180) and, due to the fact that angles are absolute, such motion is also propagated to all the affected joints via a different PDDL+ process. The effects of gravity on a joint can be stopped for two reasons: (i) the angle has reached the rest position (360/0 degrees), or (ii) the corresponding link has been grabbed by the gripper of the robot.

Uniformly Accelerated Circular Motion (MCUA). Building on top of the MCU formalisation, we introduce a more advanced representation of the impact of gravity by modelling it as a uniformly accelerated circular motion. As before, all joints angle tend to return to a 360 degree position on the z axis. However, their initial angular speed is 0, but it is uniformly accelerated. The acceleration is encoded in PDDL+ by means of an additional process, that is in charge of increasing the angular speed while the gravity

```
(:process gravity-increase
:parameters (?l1 - link)
:precondition (and  (freeToMove ?l1)
    (> (angle ?l1 ZAXES) 180)
    (< (angle ?l1 ZAXES) 360))
:effect (and
    (increase
        (angle ?l1 ZAXES) (* #t (speed-g)))
    (increasing_angle-gravity ?l1)))
```

Fig. 3: The process used in the MCU formulation to model the effect of gravity on angles between 180 and 359 degrees.

effect is active on a specific joint, and an appropriate event that "resets" the speed value when the effect of gravity ends.

As for the manipulation of angles performed by the robot, also in the MCU and MCUA formulations the effect of gravity on an angle are propagated to all the affected joints by a set of dedicated processes and events. An example of the process exploited in the MCU formulation for modelling gravity is provided in Figure 3.

3.2 Alternative Formulations

The process presented in Figure 3, as the dual `gravity-decrease` and those exploited in the MCUA formulation, has been modelled in the most human-readable way, due to the fact that robotics experts have to be involved in the modelling process. For this reason, the process keeps true a Boolean predicate, that is used to represent the fact that gravity is impacting the corresponding link. Semantically, this implies that every time step in which the process is active, the corresponding predicate is set to true. While this can be interpreted as an abuse of PDDL+ language features, it is supported by some state-of-the-art planning engines, and provides a good ground for describing and discussing the model with robotics experts.

Furthermore, in a preliminary set of experiments, we observed that some aspects of the modelling of processes and events of the presented PDDL+ models were not accepted by some of the planning engines at the state of the art. With regards to events, an example of an unaccepted event is provided in Figure 2. The `back-to-zero` event is used to reset an angle to 0 degrees as soon as the value of 360 degrees is reached. While it is easy to see that the effect of the event is making the precondition false, preventing the event to be re-applied immediately, some planning engines do not accept this formulation. Instead, they require that a Boolean precondition is falsified by the list of effects. We therefore modified the formulations with an additional predicate: it is initially set to true to allow events to be triggered. As soon as an event is triggered, its effects falsify the predicate, that is then reset to true by a subsequent `reset` event.

For the sake of completeness, and to exploit the opportunity to investigate how planning engines performance are affected by different models, we consider in our

experimental analysis both formulations. We will refer to the "Original" formulation as the formulation that is not well-supported by planning engines but maximises human readability. The other formulation will be referred to as "Modified", and it aims at maximising the parsability by planning engines. Both versions of the models, and the corresponding problem instances can be found at https://github.com/Flaudia/AMAO.

4 Experimental Analysis

This section presents the PDDL+ benchmarks and the planners employed in our analysis, as well as the results of the experiments we conducted. The main aim of this analysis is to test whether our overall PDDL+ solution can solve tasks that model practical, real-world applications, i.e., in terms of robot workspace and the number of links and physical features characterising the articulated object.

For each set of formulations, we have generated planning instances by varying the following parameters: (i) number of links of the articulated object: $3, 4, 5, 6, 7, 8, 10, 12$; (ii) in MCU: angular speed of 0.1, 0.5, and 1.0 grades per second; and (iii) in MCUA: acceleration of 0.1 and 0.5 grades per second.

In order to guarantee a fair assessment of the performance of planners according to the level of complexity used for encoding the impact of gravity, for each size of the articulated object 5 manipulation tasks were created by randomly generating initial and final configurations (cf. (5), while ensuring that a plan is possible). Those instances are then encoded in PDDL+ according to the complexity level and to the value of the corresponding MCU or MCUA parameter. Beside the size of the object, no additional parameters have to be set for the NoG formulation. Therefore, for each size of the object there are 5 tasks, encoded in 30 different problem models. The total number of problem models considered in our experimental analysis is 240.

As a reference robot we considered a Baxter dual-arm manipulator, which is widely used for research purposes and for performing manipulation tasks. This type of robot is directly supported by the presented PDDL+ formulation, and has been used –in simulation– to validate the generated plans. Moreover, it takes care also of the motion planning part. An example is shown in Figure 4.

In our analysis we have employed the following state-of-the-art PDDL+ solvers: UPMurphi [10] is possibly the most popular domain-independent PDDL+ planner, and is based on a model-checker adapted to deal with PDDL+; DiNo [22], which adds heuristics to the UPMurphi approach; and ENHSP [25,23], a numeric planner with heuristics extended to process PDDL+ problems. Those planners have been selected due to their widespread use in literature and in applications of PDDL+ planning.

Experiments have been run on a machine equipped with i7-6900K 3.20 Ghz CPU, 32 GB of RAM, running Ubuntu 16.04.3.LTS OS. 8 GB of memory were made available for each planner run, and a 5 CPU-time minutes cut-off time limit was enforced.

To be as inclusive as possible in terms of planning engines, in the rest of this section we will discuss results obtained with the Modified version of the PDDL+ models.

Tables 1-3 present the results of DiNo, ENHSP, and UPMurphi, respectively. All tables are organised as follows. The columns report the various number of links, while in the rows there are the different formulations, with their variants. For each introduced

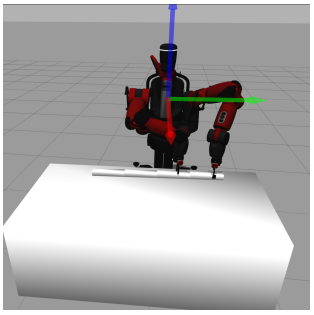

Fig. 4: A simulated Baxter robot manipulating a four link articulated object. The robot-centred reference frame is highlighted with different colours for the three reference axes.

Table 1: Results achieved by DiNo on the considered benchmarks. Sizes greater than 7 are omitted, as the planner did not solve any benchmark as well. For each dimension of the articulated object, results are presented in terms of average runtime (percentage of solved instances). Average is calculated by considering solved instances only.

		Size – Number of links of the articulated object				
		3	4	5	6	7
NoG		0.5 (40)	88.1 (40)	8.9 (20)	22.0 (40)	– (0)
MCU	0.1	0.6 (40)	130.1 (40)	13.4 (20)	27.2 (40)	– (0)
	0.5	0.6 (40)	130.2 (40)	13.5 (20)	26.8 (40)	– (0)
	1.0	0.6 (40)	125.5 (40)	13.4 (20)	26.7 (40)	– (0)
MCUA	0.1	0.6 (40)	0.9 (20)	15.4 (20)	36.2 (40)	– (0)
	0.5	0.6 (40)	1.1 (20)	15.4 (20)	36.0 (40)	– (0)

gravity encoding and number of links, it is reported the average runtime for solved instances, while in parenthesis it is reported the percentage of solved instances. In general, it is easy to derive from the results presented in the tables that, of course, the difficulty in solving the instances increases with the number of links. With regards to the level of complexity of the gravity, the NoG model –which completely ignores gravity– seems to be the easiest (relatively) to solve, followed by MCUA, while MCU seems to be the hardest. Intuitively, the fact that MCUA is easier than MCU can be due to the fact that in the MCUA model the effect of gravity slowly builds up, while in the MCU formulation the impact of gravity starts immediately at full speed.

Considering the MCU and MCUA formulations separately, the performance of the planning engines are not significantly affected by the employed parameters for MCUA, while for MCU the situation looks different: DiNo and UPMurphi do not look to be affected by the employed parameters, while the performance of ENHSP can significantly differ, also in the percentage of solved instances (see, e.g., analysis for 6 links). Our analysis suggests that this is due to the fact that DiNo and UPMurphi tend to solve only the easiest instances, for each considered size of the object, that requires short and quick to execute plans to be solved. In that, the impact of using the MCU or MCUA

Table 2: Results achieved by ENHSP on the considered benchmarks. For each dimension of the articulated object, results are presented in terms of average runtime (percentage of solved instances). Average is calculated by considering solved instances only.

		Size – Number of links of the articulated object							
		3	4	5	6	7	8	10	12
NoG		0.4 (100)	0.6 (100)	0.7 (80)	7.3 (80)	15.5 (40)	3.8 (20)	108.4 (60)	4.5 (20)
MCU	0.1	0.5 (100)	1.9 (100)	1.3 (80)	4.4 (60)	63.9 (40)	36.0 (20)	– (0)	198.5 (20)
	0.5	0.5 (100)	1.9 (100)	1.3 (80)	14.9 (80)	14.5 (40)	60.7 (20)	– (0)	– (0)
	1.0	0.5 (100)	1.7 (100)	40.0 (80)	3.2 (80)	15.4 (40)	55.3 (20)	– (0)	– (0)
MCUA	0.1	0.5 (100)	1.2 (100)	1.3 (80)	4.4 (60)	19.3 (20)	42.5 (20)	50.3 (20)	– (0)
	0.5	0.5 (100)	1.2 (100)	1.3 (80)	4.8 (60)	19.4 (20)	50.0 (20)	55.3 (20)	– (0)

Table 3: Results achieved by UPMurphi on the considered benchmarks. Sizes greater than 5 are omitted, as the planner did not solve any benchmark as well. For each dimension of the articulated object, results are presented in terms of average runtime (percentage of solved instances). Average is calculated by considering solved instances only.

		Size – Number of links		
		3	4	5
NoG		29.2 (40)	170.2 (20)	– (0)
MCU	0.1	33.1 (40)	180.3 (20)	– (0)
	0.5	32.5 (40)	178.9 (20)	– (0)
	1.0	32.1 (40)	175.6 (20)	– (0)
MCUA	0.1	2.4 (20)	– (0)	– (0)
	0.5	45.6 (40)	214.8 (20)	– (0)

model is limited. Instead, as ENHSP can solve also some more complicated instances, taking into account gravity becomes pivotal. Moreover, we can see that ENHSP can solve instances up to 12 links, while DiNo and UPMurphi stops at 6 and 2, respectively. ENHSP is also the only solver able to solve all instances up to 4 links, and the majority of the instances up to 6 links.

About the plans returned by solvers, we noticed no significant difference in terms of both quality and structure for the three approaches. It may be the case that DiNo is able to take better into account the upstream propagation of the effects of the manipulation on angles. While ENHSP tends to provide plans where the robot operates directly on the links of the joint that needs to be modified for reaching the goal position, DiNo seems to prefer the robot to work on different links, and to exploit the use of propagation processes to reach the goal position. However, given the small number of instances solved by DiNo, it is hard to assess whether this behaviour emerged by chance, or it is due to the characteristics of the planning approach exploited by the engine.

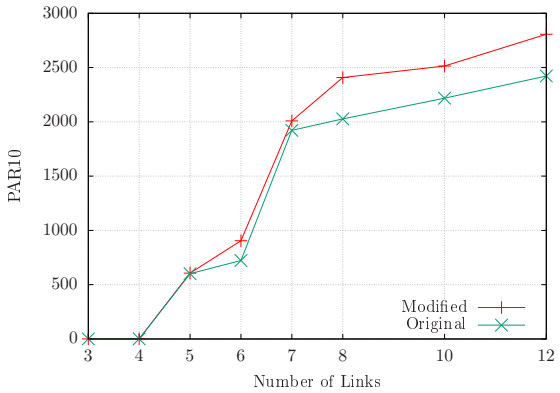

Fig. 5: Comparison of ENHSP performance, in terms of PAR10, when run on the Original formulation and on the Modified models.

Validation. The validation performed by simulating the execution of generated plans on a Baxter robot suggests that the MCU representation provides a reasonably accurate way to encode the gravity effect, as the generated plans can be executed in the simulation environment without further adjustments. An example can be found at https://github.com/Flaudia/AMAO.

4.1 Comparison of PDDL+ Models

Out of the considered planners, only ENHSP does support the Original formulation encodings, and provides valid plans. In this section we therefore exploit this planner to compare the impact of the two formulations on its performance.

Figure 5 shows how the performance of ENHSP are affected by the two sets of PDDL+ models, i.e., the Original and Modified models introduced in Section 3.

In Figure 5, performance are compared in terms of Penalised Average Runtime 10 (PAR10). PAR10 is the average runtime where unsolved instances count as 10*cutoff time. PAR10 is a metric usually exploited in machine learning and algorithm configuration techniques, as it allows to consider coverage and runtime at the same time.

In our analysis, PAR10 is calculated considering all instances of the same size, in terms of number of links, regardless of the way in which gravity is modelled. In other words, each point of the graph corresponds to the PAR10 score obtained by the planner on the 30 planning instances generated for the considered size of the object. It can be noted that when considering objects with more than 5 links, the use of the Original formulation allows to improve the performance of ENHSP. According to the Wilcoxon signed rank test [29], performed by considering all the instances, the performance improvement obtained by using the Original models is substantial and statistically significant ($p < 0.05$). In terms of quality of the generated plans, there is still no significant difference between plans generated using the Original or the Modified models.

While this may be due to the way in which the specific planner has been designed, and it is not easy to generalise, it is still interesting to note that the most human-friendly encoding is the one that allows to deliver the best performance.

5 Related Work, Conclusions and Future Work

Being able to model continuous quantities in the planning process has been subject to extensive research in Robotics, with particular reference to combined task-motion planning. In fact, combined task-motion planning may overcome those situations in which a high-level plan may not be executable in practice due to the nature of the robot workspace [5,17,27]. The approach in [5] integrates Metric-FF [16] and a sampling-based motion planner. Skolem symbols are used in [27] to incorporate workspace knowledge in predicates, which implies a symbolic-geometric mapping to be checked at runtime. Modelling the planning problem in the so-called *belief space* allows for the integration between a Markov decision process and logic-based planning [13], specifically by performing a search over finite sets of robot configurations. An interesting approach to consider is termed *Iteratively Deepened Task and Motion Planning* [9], which incorporates workspace-related information at the task planning level. Therein, task-motion interaction is done using the notion of *semantic attachment* [11], i.e., procedural functions activated during the planning process. It is noteworthy that in our case we do not consider robot motion at the continuous level in the planning process. Rather, we aim at modelling the effects of a continuous, exogenous process, i.e., gravity, on the configuration of the object the robot manipulates.

In this paper we presented two sets of PDDL+ models for the problem of robot manipulation of articulated object in a 3D workspace. Each model can then consider three levels of complexity for representing the impact of gravity. An experimental analysis, considering state-of-the-art PDDL+ planners and articulated objects of different sizes, have shown that with our PDDL+ formulations it is possible to solve tasks that model practical, real-world applications, i.e., in terms of robot workspace and the number of links and physical features characterising the articulated object. The simulation of generated plans on a Baxter dual-arm manipulator confirmed that the plans generated using the MCU formulation is a good compromize to successfully manipulate an object.

As a side effect of our work, we also designed a set of challenging PDDL+ models, that can be used to compare the performance of planning engines. We also believe they provide a good ground for testing and exploiting knowledge engineering techniques for evaluating the quality of different planning encodings [19]. As future work, we would like to test our approach on different robots models, and to evaluate more PDDL+ planners, namely SMTPlan+ [8], DReal [4] and EZCSP [2]. Both SMTPlan+ and Dreal rely on an SMT module for solving problems described using PDDL+, while EZCSP exploits Constraint Answer Set Programming (CASP) [20] encodings and solvers. SMTPlan+ has not been evaluated given that we were not able to compile it under the machines at our disposal, while EZCSP is not included because currently it does not directly support PDDL+, and the problem needs to be expressed in CASP. Finally, DReal does not seem to be capable of generating a plan for any of the models.

References

1. Agostini, A., Torras, C., Worgotter, F.: Integrating task planning and interactive learning for robots to work in human environments. In: Proc. of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011). pp. 2386–2391. IJCAI/AAAI (2011)
2. Balduccini, M., Magazzeni, D., Maratea, M., Leblanc, E.: CASP solutions for planning in hybrid domains. Theory and Practice of Logic Programming **17**(4), 591–633 (2017)
3. Bodenhagen, L., Fugl, A., Jordt, A., Willatzen, M., Andersen, K., Olsen, M., Koch, R., Petersen, H., Kruger, N.: An adaptable robot vision system performing manipulation actions with flexible objects. IEEE Transactions on Automation Science and Engineering **11**(3), 749–765 (2014)
4. Bryce, D., Gao, S., Musliner, D.J., Goldman, R.P.: SMT-based nonlinear PDDL+ planning. In: Proc. of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015). pp. 3247–3253. AAAI Press (2015)
5. Cambon, S., Alami, R., Gravot, F.: A hybrid approach to intricate motion, manipulation and task planning). The International Journal of Robotics Research **28**(1), 104–126 (2009)
6. Capitanelli, A., Maratea, M., Mastrogiovanni, F., Vallati, M.: Automated planning techniques for robot manipulation tasks involving articulated objects. In: Proc. of the XVIth International Conference of the Italian Association for Artificial Intelligence. Lecture Notes in Computer Science, vol. 10640, pp. 483–497. Springer (2017)
7. Capitanelli, A., Maratea, M., Mastrogiovanni, F., Vallati, M.: On the manipulation of articulated objects in human–robot cooperation scenarios. Robotics and Autonomous Systems **109**, 139 – 155 (2018)
8. Cashmore, M., Fox, M., Long, D., Magazzeni, D.: A compilation of the full PDDL+ language into SMT. In: Proc. of the 26th International Conference on Automated Planning and Scheduling (ICAPS 2016). pp. 79–87. AAAI Press (2016)
9. Dantam, N., Kingstone, Z., Chaudhuri, S., Kavraki, L.: An incremental constraint-based framework for task and motion planning). The International Journal of Robotics Research **37**(10), 1134–1151 (2018)
10. Della Penna, G., Magazzeni, D., Mercorio, F., Intrigila, B.: Upmurphi: A tool for universal planning on PDDL+ problems. In: Proc. of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009). AAAI (2009)
11. Dornhege, C., Eyerich, P., Keller, T., Trug, S., Brenner, M., Nebel, B.: Semantic attachments for domain-independent planning systems. In: Proc. of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009). AAAI (2009)
12. Fox, M., Long, D.: Modelling mixed discrete-continuous domains for planning. Journal of Artificial Intelligence Research **27**, 235–297 (2006)
13. Garrett, C., Perez, T., Kaelbling, L.: FF-rob: leveraging symbolic planning for efficient task and motion planning. The International Journal of Robotics Research **37**(1), 104–136 (2018)
14. Henrich, D., Worn, H.: Robot manipulation of deformable objects. Advanced Manufacturing, Springer (2000)
15. Heyer, C.: Human-robot interaction and future industrial robotics applications. In: Proc. of IEEE International Conference on Intelligent Robots and Systems (IROS 2010). pp. 4749–4754. IEEE (2010)
16. Hoffmann, J.: The Metric-FF planning system: translating ́ignoring delete lists ́to numeric state variables. Journal of Artificial Intelligence Research **20**, 291–341 (2003)
17. Kaelbling, L., Perez, T.: Integrated task and motion planning in the belief space. The International Journal of Robotics Research **32**(9-10), 1194–1227 (2013)
18. Krüger, J., Lien, T., Verl, A.: Cooperation of humans and machines in the assembly lines. CIRP Annals - Manufacturing Technology **58**(2), 628–646 (2009)

19. McCluskey, T.L., Vaquero, T.S., Vallati, M.: Engineering knowledge for automated planning: Towards a notion of quality. In: Proceedings of the Knowledge Capture Conference (K-CAP 2017). pp. 14:1–14:8 (2017)
20. Mellarkod, V.S., Gelfond, M., Zhang, Y.: Integrating answer set programming and constraint logic programming. Annals of Mathematics and Artificial Intelligence **53**(1-4), 251–287 (2008)
21. Nair, A., Chen, D., Agraval, P., Isola, P., Abbeel, P., Malik, J., Levine, S.: Combining self-supervised learning and imitation for vision-based rope manipulation. In: Proc. of the 2017 IEEE International Conference on Robotics and Automation (ICRA 2017). pp. 2146–2153. IEEE (2017)
22. Piotrowski, W.M., Fox, M., Long, D., Magazzeni, D., Mercorio, F.: Heuristic planning for PDDL+ domains. In: Proc. of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016). pp. 3213–3219. IJCAI/AAAI Press (2016)
23. Ramírez, M., Papasimeon, M., Lipovetzky, N., Benke, L., Miller, T., Pearce, A.R., Scala, E., Zamani, M.: Integrated hybrid planning and programmed control for real time UAV maneuvering. In: Proc. of the 17th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2018). pp. 1318–1326. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM (2018)
24. Saadat, M., Nan, P.: Industrial applications of automatic manipulation of flexible materials. Industrial Robot: an International Journal **29**(5), 434–442 (2002)
25. Scala, E., Haslum, P., Thiébaux, S., Ramírez, M.: Interval-based relaxation for general numeric planning. In: Proc. of the 22nd European Conference on Artificial Intelligence (ECAI 2016). Frontiers in Artificial Intelligence and Applications, vol. 285, pp. 655–663. IOS Press (2016)
26. Schulman, J., Ho, J., Lee, C., Abbeel, P.: Learning from demonstrations through the use of non-rigid registration. In: M. Inaba and P. Corke (Eds.) Robotics Research, Springer Tracts in Advanced Robotics, vol. 114. Springer International Publishing (2016)
27. Srivastava, S., Fang, E., Riano, L., Chitnis, R., Russell, S., Abbeel, P.: Combined task and motion planning through an extensible planner-independent interface layer. In: Proc. the 2014 IEEE International Conference on Robotics and Automation (ICRA 2014). pp. 639–646. IEEE (2014)
28. Wakamatsu, H., Arai, E., Hirai, S.: Knotting and unknotting manipulation of deformable linear objects. International Journal of Robotic Research **25**(4), 371–395 (2006)
29. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin **1**(6), 80–83 (1945)
30. Yamakawa, Y., Namiki, A., Ishikawa, M.: Dynamic high-speed knotting of a rope by a manipulator. International Journal of Advanced Robotic Systems **10**, 1–12 (2013)