Automated Planning Techniques for Robot Manipulation Tasks Involving Articulated Objects

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

[1] DIBRIS, Univ. degli Studi di Genova, Viale F. Causa 15, 16145 Genova, Italy
name.surname@unige.it
[2] University of Huddershield, West Yorkshire, HD1 3DH, United Kingdom.
M.Vallati@hud.ac.uk

Abstract. The goal-oriented manipulation of articulated objects plays an important role in real-world robot tasks. Current approaches typically pose a number of simplifying assumptions to reason upon how to obtain an articulated object's goal configuration, and exploit *ad hoc* algorithms. The consequence is two-fold: firstly, it is difficult to generalise obtained solutions (in terms of actions a robot can execute) to different target object's configurations and, in a broad sense, to different object's physical characteristics; secondly, the representation and the reasoning layers are tightly coupled and inter-dependent.
In this paper we investigate the use of automated planning techniques for dealing with articulated objects manipulation tasks. Such techniques allow for a clear separation between knowledge and reasoning, as advocated in Knowledge Engineering. We introduce two PDDL formulations of the task, which rely on conceptually different representations of the orientation of the objects. Experiments involving several planners and increasing size objects demonstrate the effectiveness of the proposed models, and confirm its exploitability when embedded in a real-world robot software architecture.

1 Introduction

The manipulation of non rigid objects, including *articulated* or *flexible* objects, such as strings, ropes or cables, is one of the most complex tasks in Robotics [20, 11]. Apart from issues related to grasping and dexterity, and differently from rigid objects, the *configuration* of an articulated or flexible object (i.e., the set of relative *poses* of its constituent parts) varies due to the relative position of its constituent parts with respect to each other. This induces a representation problem for such objects, which, on the one hand, is tightly connected with robot perception capabilities and their accuracy, and on the other hand impacts on processes reasoning about configuration changes and the associated robot manipulation actions.

In the literature about the manipulation of non rigid objects, this problem did not receive sufficient attention, nor a principled formalisation is available. Indeed, it is possible to find examples in which robots exhibit the capability of manipulating ropes [25], tying or untying knots [19] and operating on mobile parts of the environment, such as handles of different shapes [5], home furniture [12] or valves in search and rescue settings [17]. However, in all these cases, manipulation actions are directly grounded on perceptual cues, such as the peculiar geometry of the object to deal with [1], assumed to be easy to identify in a robust way, or based on *a priori* known or learned information about the object to manipulate, e.g., its stiffness or other physical features [7, 6]. As a result, every time either the element that has to be manipulated or the manipulator changes, a new reasoner has to be developed from scratch.

A structured approach to perception, representation and reasoning, as well as execution, seems beneficial: on the one hand, we can decouple perception and representation issues, thus not being tied to specific perception approaches or *ad hoc* solutions; on the other hand, domain knowledge and reasoning logic can be separated, with the advantages of an increased maintainability, and the possibility to interchange reasoners and models in a modular way.

In this paper, we investigate the use of automated planning techniques for manipulation tasks involving articulated objects. Such techniques assume an abstract model of the object to manipulate, a clear separation between knowledge representation and reasoning, and the use of standard languages, such as PDDL [16], and widely available domain-independent planners. Language standard and planners' efficiency have been fostered by the International Planning Competition series (see, e.g., [22]). Our contribution is at the problem formalisation and modelling levels. It should be noted that the representation of an articulated object can be modelled using two alternative approaches, which differ on how link orientations are expressed: *relative*, with respect to each other (e.g., any link with respect to the previous one, assuming an ordering among links), or *absolute*, with respect to an external, possibly robot-centered, reference frame. In this context, it is evident that we are not trying to generate joint trajectories to achieve a desired object configuration, but rather determine a series of model-defined actions towards such goal. Roboticists are familiar with numerical methods and motion control strategies for articulated structures, what the planner should provide is a series of intermediate reference joint states to be fed to those lower level systems.

From a robotics perspective, a relative representation is very sensitive to perception issues: *small* perception errors in link orientations can lead to dramatically *huge* errors in the estimate of the object's configuration, since it is necessary to compute forward all the robot-centred orientations to support manipulation actions: this seems to suggest that an absolute representation would be preferable, but also the efficiency of planners on the related formulations must be taken into account.

In this respect, starting from the relative and absolute representations of link orientations, we propose two planning models: the first, which we refer to as *basic*, assumes pairwise relative link orientations and primary features of PDDL; the second, which is termed *conditional*, treats orientations as absolute and employs also conditional statements. Experiments have been performed using an architecture integrating a modified ROSPlan framework [4] on a dual-arm robot manipulator. They show that our approach (i) efficiently solves tasks with a realistic size, in terms of number of links constituting the object and the resolution of their orientations, and (ii) scales in a satisfactory way with increasingly more complex problems, which is of particular relevance since it provides a challenging benchmark for the planning community, and can be seen as an important step toward the manipulation of flexible objects.

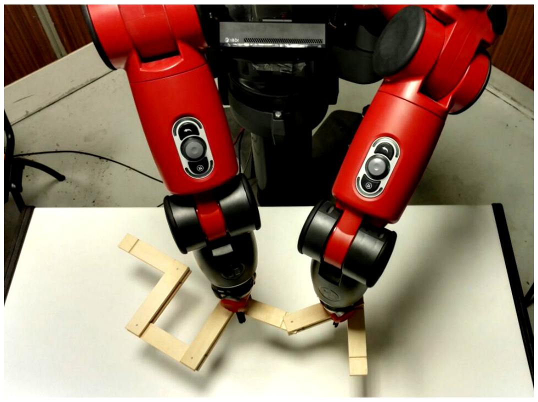

Fig. 1. The reference robotic framework.

The paper is structured as follows. Section 2 provides the reader with needed preliminaries about our scenario and automated planning, whereas Section 3 introduces the problem statement. Then, Section 4 presents the two models, whose evaluation with automated planners is shown in Section 5. Conclusions follow.

2 Background

In this section we provide the necessary background on the reference scenario, and on automated planning.

2.1 The Reference Scenario

The tabletop scenario we consider involves a Baxter dual-arm robot manipulator from Rethink Robotics (see Figure 1). Each arm has 7 degrees of freedom and is equipped with a standard gripper. An RGB-D device located on the Baxter's *head* and pointing downward is used to perceive the robot's frontal workspace. The workspace is constituted by a table, on which articulated objects can be manipulated by rotating its constituting parts. Given our reference scenario, the granularity of rotations can not be small. We employ wooden objects, which have been purposely hand crafted to minimise perceptual errors: the first has three $40\ cm$ long links (and two loose joints), whereas the second has seven $20\ cm$ long links (and six stiff joint). On the second object we fixed a QR tag to each link, in order to quickly determine its orientation.

A software architecture has been developed, based on the well known ROS framework and integrating ROSPlan, which allows sensory-based knowledge representation, action planning and execution via a number of nested control loops. A point cloud or link poses (determined using QR tags) are obtained either continuously or *on demand* from the RGB-D device. Perception data are processed in order to obtain a model-based representation of the scene [3], for instance the configuration of an articulated object, which is maintained within an OWL-based ontology [13]. The ontology is updated whenever a new perception is available. In order to obtain a new object's goal configuration, the information within the ontology can be accessed by a planner, which can extract information to build the initial state of the planning problem. Since the planner is treated as a ROS service, any suitable planner can be used as long as it adheres to a well-defined communication interface. If a plan is found, each manipulation action therein is executed. After each execution a new perception is obtained, whereas the scene representation in the ontology is updated, and compared with the expected effects of the action: if they are compatible, the execution continues, otherwise re-planning occurs. The execution continues until the final state of the plan is reached, or aborted as per designer's instructions.

2.2 Automated Planning

Automated planning, and specifically classical planning, deals with finding a (partially or totally ordered) sequence of actions, which modify a static, deterministic and fully observable *environment* from an initial state to a desired goal state [9].

In classical planning, the environment is represented as an appropriate set P of $|P|$ First Order Logic *predicates*, $p_1, \ldots, p_{|P|}$, whereas *states* $s_1, \ldots, s_{|S|}$ are defined as appropriate sets of ground predicates $\bar{p}_1, \ldots, \bar{p}_{|\bar{P}|}$.
An *operator* $o = \big(name(o), pre(o), eff^-(o), eff^+(o)\big)$ is defined such that $name(o) = o\_name(x_1, \ldots, x_K)$, where o_name is a unique operator name and $x_1, \ldots, x_K$ are its K arguments, $pre(o)$ is the set of predicates P_{pre} representing the operator's preconditions, whereas $eff^-(o)$ and $eff^+(o)$ are, respectively, the sets of predicates P_{eff^-} and P_{eff^+} representing the operator's negative and positive effects. As a consequence, *actions* $a_1, \ldots, a_{|A|}$ are ground instances of planning operators. An action $a = \big(pre(a), eff^-(a), eff^+(a)\big)$ is *applicable* in a state s if and only if $pre(a) \subseteq s$. If allowed, the application of a in s results in a new state such that $\big(s \setminus eff^-(a)\big) \cup eff^+(a)$.

A *planning domain* $\mathcal{D}$ is specified via sets of predicates $P_{\mathcal{D}}$ and operators $O_{\mathcal{D}}$, such that $\mathcal{D} = (P_{\mathcal{D}}, O_{\mathcal{D}})$. A *planning problem* $\mathcal{P}$ is specified via a planning domain $\mathcal{D}$, an initial state s_i and set of goal atoms $\bar{P}$ (both made up of ground predicates), such that $\mathcal{P} = \big(\mathcal{D}, s_i, \bar{P}\big)$. A *solution plan* $\mathcal{S}$ is a sequence of I actions $a_1, \ldots, a_I$ such that, starting from the initial state s_i, a consecutive application of the actions in the plan results in a final state s_f that satisfies the goal $\bar{P}$.

3 Problem Statement

In general terms, the problem we consider in this paper can be stated as follows: given an articulated object, determine a solution plan that modifies the initial object's configuration to a specified goal configuration, where the solution plan is made up of a number of manipulation actions to be executed by a robot. In order to better specify the boundaries of the problem we consider, let us pose the following assumptions:

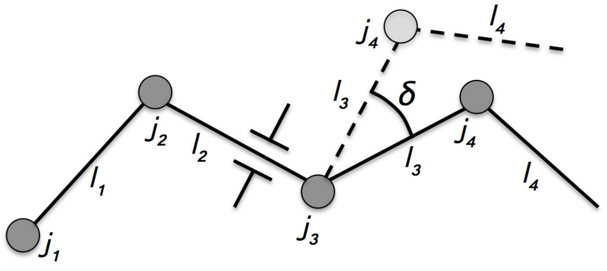

Fig. 2. An example of rotation: l_3 rotates around j_3 of an angle δ while l_2 is kept still, which induces l_4 to rotate.

1. We consider articulated objects as *simplified* models for fully flexible objects. These can be modelled as articulated objects with a huge number of links and joints. This assumption is widely accepted [25].
2. We do not consider the effects of gravity on the object being manipulated, nor those of any external force but manipulation actions. For this reason, the object is considered as laying on a horizontal plane (large enough to accommodate it). As a consequence, we consider articulated objects which lay strictly on the plane.
3. We assume sensing and representation to be decoupled, the latter assuming *perfect* sensing. On the basis of the features extracted from sensing data, this leads to different problem formulations.
4. The object can be easily manipulated by standard Baxter's grippers. The specific object we use has been purposely manufactured to that aim, and therefore we do not consider issues related to grasping or dexterity.

More formally, an articulated object α is defined as an ordered set $\mathcal{L}$ of $|L|$ links and an ordered set $\mathcal{J}$ of $|J|$ joints, such that $\alpha = (\mathcal{L}, \mathcal{J})$. Each link l is characterised by a length λ_l and an orientation θ_l on the plane, whose meaning depends on the considered planning model, namely basic or conditional. Therefore, a configuration C_α is a $|L_\alpha|$-ple such that $\left(\theta_{l_1}, \ldots, \theta_{l_{|L|}}\right)$, i.e., the orientations of all the links. Obviously enough, since $\mathcal{L}$ and $\mathcal{J}$ are ordered sets, links and joints are pairwise correlated, such that a link l_l is bounded upstream by j_l and downstream by j_{l+1}, apart from $l_{|L|}$.

Configurations change as a consequence of manipulation actions. In our case, we only consider rotations of a given link l_l around the corresponding joints j_l or j_{l+1}. If we refer to δ as the *granularity* associated with variations in orientations, then a manipulation action a operating on a link l_l can either increase or decrease its orientation θ_l by δ, with respect to an axis perpendicular to the horizontal plane. In doing so, and given the stiffness of α, the robot needs the use of two grippers: the first is used to keep the upstream (resp. downstream) l_{l-1} (resp. l_{l+1}) link still, which is a non modelled action in the solution plan (nonetheless executed by the robot when needed), whereas the second is used to rotate l_l around j_l (resp. j_{l+1}).

An example can be found in Figure 2, where a 4-link, 4-joint articulated object is shown. The initial state s_i corresponds to link poses in black, whereas in the final state s_f links l_3 and l_4 must be rotated by δ, i.e., l_3 must rotate around j_3. The expected sequence of manipulation actions includes: grasping and keeping l_2 still, grasping l_3, rotating l_3 counter clockwise around j_3 of about δ, releasing l_3 and releasing l_2. However, the first action, which is a necessary prerequisite for the rotation to occur, need not to be explicitly modelled, but can be delegated to the robot action execution system.

4 Proposed Formulations

In order to tackle the problem introduced above, and to evaluate the two possible semantics associated with link orientations, we designed two PDDL formulations, which exploit different sets of language features. The *basic* formulation employs the `:STRIPS` subset of PDDL, extended with `equalities` and `negative-preconditions`, whereas the *conditional* version requires also the use of `conditional-effects`. Notably, the precision limits of most manipulators requires the granularity discretisation of angular movements, hence there is no practical necessity for continuous or hybrid planning models. Therefore, PDDL provides an appropriate level of abstraction.

On the one hand, in the basic formulation, given a link l_l, its orientation θ_l must be considered as being relative, for instance, to the orientation of the upstream link l_{l-1}. From a planning perspective, each manipulation action changing θ_l does not affect any other upstream or downstream link orientations, since all of them are relative to each other, and therefore the planning process is computationally less demanding. However, since manipulation actions are expected to be based on link orientations grounded with respect to a robot-centred reference frame, i.e., absolute in terms of pairwise link orientations, a conversion must be performed, which may be greatly affected by perceptual noise, therefore leading to inaccurate or even inconsistent representations. On the other hand, in the conditional formulation, θ_l is considered as absolute, and therefore it can be associated directly with robot actions. Unfortunately, this means that each manipulation action changing θ_l does affect numerically *all* other upstream or downstream link orientations, depending on which side of the object is kept still, in the representation, which must be kept track of using conditional effects in the planning domain.

It is noteworthy that the use of advanced PDDL features, such as conditional effects, may allow for a more accurate representation of the domain but, at the same time, it may reduce the number of planners able to reason on the model.

4.1 Basic Formulation

As described in Section 3, an articulated object α is represented using two ordered sets of joints and links. We use a `connected` predicate to describe the sequence of links in terms of binary relationships involving a joint j_{l+1} and a link l_l, which induces a pairwise connection between two links, namely l_l itself and l_{l+1}, since they share the same joint j_{l+1}. We assume that each joint j_l is associated with an angle θ_j, which ranges between 0 and 359 deg, through the predicate `angle-joint`. Obviously enough, in such a range it holds that $\theta_j = \theta_{l+1} - \theta_l$. As we anticipated, this formulation assumes that link orientations are expressed as pairwise relative to each other. This means that the robot perception system is expected to provide the representation layer with the set of joint angles $\theta_1, \ldots, \theta_{|J|}$ as primitive information, whereas the set of link orientations

```
(:action increase-angle
   :parameters (?link1 ?link2 - link
   ?joint - joint ?a1 ?a2 - angle)
   :precondition (and
     (connected ?joint ?link1)
     (connected ?joint ?link2)
     (not (= ?link1 ?link2))
     (angle-joint ?a1 ?joint)
     (angle-before ?a1 ?a2))
   :effect (and
     (not (angle-joint ?a1 ?joint))
     (angle-joint ?a2 ?joint)))
```

Fig. 3. The *basic* formulation of `increase-angle`.

$\theta_1, \ldots, \theta_{|L|}$ is not directly observable, but must be computed applying forward kinematics formulas to the object's configuration C_α. As we discussed already, if noise affects the perception of joint angles, as it typically does, the reconstruction of the object's configuration may differ from the real one, and it worsens with link lengths. This position significantly simplifies the planning model's complexity: from a planner's perspective, the modification of any link orientations does not impact on other relative joint angles, and therefore manipulation actions can be unfolded *in any order* the planner deems fit.

Angles are specified using *constants*, which are then ordered using the `angle-before` predicate. The difference between constant values is the *granularity* δ of the resolution associated to modelled orientations. For example, `d45` and `d90` are used as constants representing, respectively, a 45 and a 90 deg angle. Then, a predicate `(angle-before d45 d90)` is used to encode the fact that `d45` is the granularity step preceding `d90`, i.e, in this case $\delta = 45$ deg.

The domain model includes two planning operators, namely `increase-angle` (shown in Figure 3) and `decrease-angle`. Intuitively, the former can be used to increase the angle of a selected joint of a δ step, while the latter is used to decrease the joint's angle, by operating on the two connected links. As an example, if $\delta = 45$ deg and a joint angle $\theta_j = 135$ deg, `increase-angle` would produce $\theta_j = 180$ deg, whereas `decrease-angle` $\theta_j = 90$ deg. In the operator's definition, `?link1` and `?link2` represent any two links l_l and l_{l+1}, `?joint` is the joint j_{l+1} between them, whereas `?a1` and `?a2` are the current and the obtained joint angles, respectively. If `?joint` connected two different links `?link1` and `?link2`, the angle `?a1` of such joint would be increased of a δ step and become `?a2`. A similar description could be provided for `decrease-angle`.

A problem is defined by specifying initial and final states. The former includes the *topology* of the articulated object in terms of `connected` predicates, and its initial configuration using `angle-joint` predicates; the latter describes its goal configuration using relevant `angle-joint` predicates.

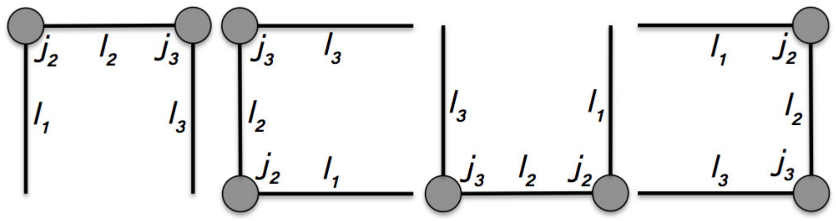

Fig. 4. Without end point joints, the basic formulation cannot discriminate among these four configurations.

It is noteworthy that, as shown in Figure 2, we add one seemingly unnecessary joint to the configuration, as one of the *end points* of the link chain. As a matter of fact, from a representation perspective, the use of relative angles leads to issues in discriminating between some configurations of the articulated object. Let us consider, for instance, the case of an articulated object made up of 3 links, namely l_1, l_2 and l_3, which are connected by two joints, namely j_2 (connecting l_1 and l_2) and j_3 (connecting l_2 and l_3). Then, let us set both joint angles to $90\ deg$. If j_2 and j_3 were treated as relative, in the planning process it would be impossible to distinguish between configurations $C_{\sqcap}$, $C_{\sqsubset}$, $C_{\sqcup}$ and $C_{\sqsupset}$ in Figure 4. In order to deal with this drawback, the end point joint j_1 and a related "hidden" link l_0 (not shown in the Figure) can be added to one of the articulated object's extremes. This hidden link defines an *ad hoc* reference frame that allows for discriminating among configurations characterised by the same shape, but with different orientations. Such hidden links must be added to problem definitions.

4.2 Conditional Formulation

The *conditional* formulation differs from the *basic* one in that joint angles θ_j originate from link orientations expressed with respect to a unique, typically robot-centred, reference frame, and as such are absolute. Therefore, the set of link orientations $\theta_1, \ldots, \theta_{|L|}$ is assumed to be directly observable by the robot perception system. However, if a manipulation action is planned, which modifies a given joint angle θ_j, not only the related link orientations θ_l or θ_{l+1} (depending on whether the upstream or downstream link is kept still) must be updated, but it is necessary to propagate such changes to all the upstream or downstream link orientations. As a consequence, such a representation increases the complexity of the planning tasks but is more robust to perception errors: in fact, perceiving independent link orientations induces an *upper bound* on the error associated with their inner angle.

The `connected`, `angle-joint` and `angle-before` predicates are the same as in the *basic* formulation, subject to the different semantics associated with joint angles. Also in the *conditional* formulation two planning operators are used, namely `increase-angle` (shown in Figure 5) and `decrease-angle`. However, with respect to the *basic* formulation, the effects of the operator differ. In particular, the model assumes that we can represent which joints are affected when a link is rotated around one of its associated joints. This is done using the `affected` predicate, i.e., a ternary

```
(:action increase-angle
   :parameters (?link1 ?link2 - link
   ?joint - joint ?a1 ?a2 - angle)
   :precondition (and
     (connected ?joint ?link1)
     (connected ?joint ?link2)
     (not (= ?link1 ?link2))
     (angle-joint ?a1 ?joint)
     (angle-before ?a1 ?a2))
   :effect (and
     (not (angle-joint ?a1 ?joint))
     (angle-joint ?a2 ?joint)
     (forall (?js - joint ?a3 ?a4 - angle)
       (when (and
       (affected ?js ?link1 ?joint)
       (not (= ?js ?joint))
       (angle-joint ?a3 ?js)
       (angle-before ?a3 ?a4) )
       (and
          (not (angle-joint ?a3 ?js))
          (angle-joint ?a4 ?js)))))
```

Fig. 5. The *conditional* version of `increase-angle`.

predicate `(affected ?joint1 ?link1 ?joint2)`, where `?link1` is the link that is rotated, `?joint2` is the joint around which `?link1` rotates, and `?joint1` is a joint affected by this rotation. So, if `?joint1` were affected, its angle would be changed as well in the conditional statement and, as such, it would affect other joints via its corresponding link.

With reference to `increase-angle`, as in the previous case, the joint angle `?joint`, located between `?link1` and `?link2`, is increased by δ, according to the `angle-before` predicate. If rotating `?link2` around `?joint` affects `?js`, the latter is updated and affects in cascade all other joints upstream or downstream.

In terms of problem definitions, it is necessary to include the list of appropriately defined `affected` predicates.

5 Experimental Evaluation

The aim of this experimental evaluation is to assess the computational performance of the *basic* and *conditional* models, and in particular whether the proposed planning-based approach can be effective in a real-world robot software architecture. First we discuss the experimental settings, then we show how the two planning models scale with increasingly difficult problems (in terms of the number of joints in the articulated object and the granularity δ associated with link orientations).

Settings. We selected 4 planners, based on their performance in the agile track of the 2014 International Planning Competition: Madagascar (Mp) [18], Probe, SIW [15], and Yahsp3 [23]. We also included Lpg [8] due to its widespread use in real-world planning applications. Both Yahsp3 and Lpg do not support conditional effects.

Experiments have been performed on a workstation equipped with 2.5 Ghz Intel Core 2 Quad processors, 4 GB of RAM and the Linux operating system.

Synthetic problem instances have been randomly generated parameterised on the number of joints j^* (the same as the number of links) and the number of allowed orientations g^* (which induces certain granularity values), which both affect the problem's size. Once j^* and g^* are defined, a configuration C^* is determined by normally sampling, for each link, among the finite set of possible orientations induced by g^*. The cutoff time has been set to 300 CPU-time seconds. Generated plans have been validated using the well-known VAL tool [10], in order to check their correctness with respect to the planning models, and also to verify the presence of flaws.

Computational Performance. In a first series of tests we analyse the sensitivity of planning models with respect to increasing values for j^*. It is noteworthy that in the literature there is hardly evidence of objects manipulated by robots made up of more than four or five joints at best. However, it seems reasonable to model flexible objects as articulated objects with a huge number of links. In that case, the higher the number of links, the better the approximation given by the model. We generated problem instances in which j^* ranges between 3 to 20, fixing $g^* = 90\ deg$. As such, only four orientations are possible. For each value of j^*, three instances have been generated and the results averaged.

Unsurprisingly, the *basic* model is faster than the *conditional* model. When using the *basic* model, j^* has no significant impact on a planner running time. In all cases, valid plans have been found in less than 1 CPU-time second. This is due to the fact that, in the *basic* model, each rotation is independent, and the final state can be quickly reached by focusing on one joint at a time. Instead, the *conditional* model requires the planner to consider the impact a rotation has on other links. Moreover, the size of the *conditional* problem model exponentially increases with j^*, due to the presence of `affected` predicates that need to be specified.

Figure 6 shows, for the three planners able to cope with conditional effects, how the average runtime is affected by j^*. SIW is the most negatively affected, while Mp and Probe share a similar trend. According to the Wilcoxon test [24], the performance of SIW is statistically significantly worse than those of Mp and Probe. Furthermore, Probe has statistically better performance than Mp. None of the planner solved, within the 300 CPU-time seconds cutoff, any instance with j^* greater than 15.

When assessing the quality of plans generated using the *basic* model, empirical evidence indicates that Mp, Probe and SIW generate plans of similar length (approx. 10 actions). Yahsp3 and Lpg find plans consistently worse than those found by the above mentioned planners: on average, LPG (Yasp3) finds plans that are 85% (25%) longer than those found by Mp, Probe and SIW. Instead, when using the *conditional* model, SIW typically provides the shortest plans on average, followed by Probe (17% longer) and Mp (50% longer). Interestingly, when comparing the quality of plans generated by the same planner, on the two considered models, we can identify two different be-

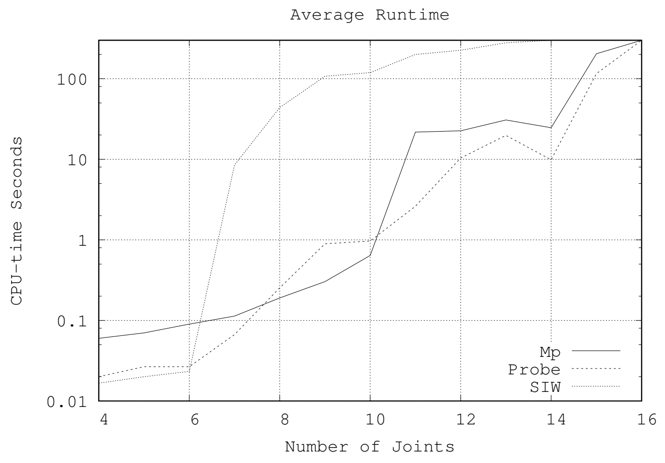

Fig. 6. Average CPU-time seconds needed by Mp, Probe, and SIW to solve instances with j^* from 4 to 16 using the *conditional* model.

haviours. Mp and Probe seem to suffer the way conditional effects propagate rotations: plans generated with the *conditional* model are significantly longer. An in-depth analysis suggests that such planners use many actions to *fix* joint angles affected by previous manipulation actions. On the contrary, SIW seems to exploit the additional information leveraged by conditional effects, and this is reflected by shorter plans.

In a second series of tests we investigate how g^* affects a plan's feasibility and size. Intuitively, a high value for g^* (e.g., 90 deg) implies a low number of possible orientations (e.g., 4), thus a small number of manipulation actions must be planned, and *viceversa*. We generated problem instances in which j^* is set to 5, 10, 15, or 20. For each size, we modelled the problem with three different values for g^*, namely 90 deg, 60 deg and 30 deg, thus leading to 4, 6 and 12 possible orientations. For each value of j^* and g^*, three instances have been generated and the results averaged.

As expected, also in this case, the *basic* model allows all the planners to generate solution plans. As in the previous case, all instances are solved in less than 1 CPU-time second. When considering the *conditional* model, the observed results show that Probe has the best scalability. SIW and Mp run out of time also for small values of j^*, when $g^* = 30\ deg$. Figure 7 presents the impact of the considered values for g^* on the average runtime performance of Probe. It is apparent that the granularity has a very strong impact on planning performance: on instances with the same number of joints, the runtime can increase up to two orders of magnitude for different granularities in link orientations.

Finally, we executed some of the plans obtained by Probe when considering objects with j^* of 3 and 7 and $g^* = 90\ deg$, on the actual Baxter manipulator shown in Figure 1, which was controlled using the ROSPlan framework [4]. All plans were successfully executed by the robot, which was able to manipulate the object in order to provide the required goal configuration.

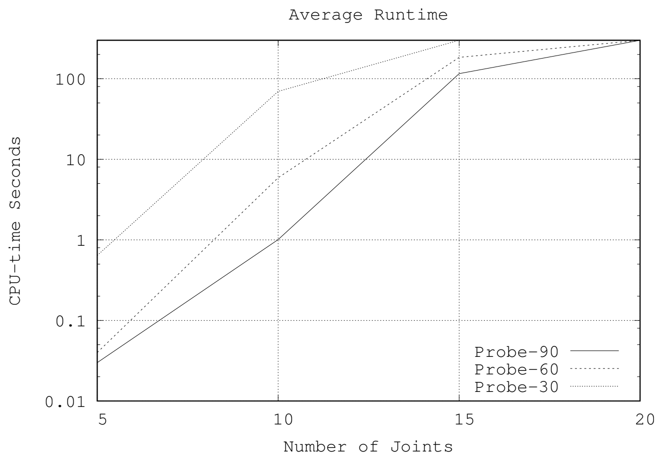

Fig. 7. Average CPU-time seconds needed by Probe to solve instances with j^* equal to 5, 10, 15, or 20, and g^* equal to 90, 60, and 30 deg, using the *conditional* model.

6 Conclusion

This paper presented the use of automated planning to plan for robot manipulation of articulated objects, which is one of the most complex tasks in robotics. We introduced two PDDL models, based either on *relative* or *absolute* representation of links orientation. The former model requires the robotic framework to be able to provide an accurate perception of the position of the articulated object, while the latter is more robust to perception errors, at the price of a higher complexity.

Our experimental analysis, which involved four state-of-the-art planning engines and objects of different sizes, performed on synthetically generated problem instances and by using the ROSPlan framework for controlling a Baxter manipulator, shows that: (i) both models allow to generate plans that are executable by a typical robotics manipulator; (ii) both the models allows considered planners to efficiently solve tasks with a realistic size; and (iii) the basic model can be fruitfully exploited when very large objects are considered, as an approximation of flexible objects. We also observed that the conditional model can be exploited as a challenging benchmarks for testing the capabilities of planning engines: the number of conditional effects per grounded action can be extremely large.

We see several avenues for future work. We plan to investigate how to extend the proposed models in order to represent multi-dimensional joint movements. We are also interested in testing the proposed approach using different manipulators, and to test its feasibility for representing flexible –rather than articulated– objects such as ropes. Finally, we also envisage to study knowledge processing mechanisms to possibly improving the flexibility of control processes (see, e.g., [14, 2, 21]).

References

1. Berenson, D.: Manipulation of deformable objects without modelling and simulating deformation. In: Proceedings of the 2013 IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS 2015). Tokyo, Japan (November 2013)
2. Borgo, S., Cesta, A., Orlandini, A., Umbrico, A.: A planning-based architecture for a reconfigurable manufacturing system. In: Coles, A.J., Coles, A., Edelkamp, S., Magazzeni, D., Sanner, S. (eds.) Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling, ICAPS 2016. pp. 358–366. AAAI Press (2016)
3. Buoncompagni, L., Mastrogiovanni, F.: A software architecture for object perception and semantic representation. In: Proceedings of the Second Italian Workshop on Artificial Intelligence and Robotics (AIRO 2015). Ferrara, Italy (September 2015)
4. Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carrera, A., Palomeras, N., Hurtós, N., Carreras, M.: Rosplan: Planning in the robot operating system. In: Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS. pp. 333–341 (2015)
5. Dang, H., Allen, P.: Robot learning of everyday object manipulations via human demonstrations. In: Proceedings of the 2010 IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS 2010). Taipei, Taiwan (October 2010)
6. Elbrechter, C., Haschke, R., Ritter, H.: Folding paper with anthropomorphic robot hands using real-time physics-based modeling. In: Proceedings of the 2012 IEEE-RAS International Conference on Humanoid Robotics (HUMANOIDS 2012). Osaka, Japan (October 2012)
7. Frank, B., Schmedding, R., Stachniss, C., Teschner, M., Burgard, W.: Learning the elasticity parameters of deformable objects with a manipulation robot. In: Proceedings of the 2010 IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS 2010). Taipei, Taiwan (October 2010)
8. Gerevini, A.E., Saetti, A., Serina, I.: Planning through stochastic local search and temporal action graphs in LPG. Journal of Artificial Intelligence Research 20, 239–290 (2003)
9. Ghallab, M., Nau, D., Traverso, P.: Automated planning, theory and practice. Morgan Kaufmann Publishers (2004)
10. Howey, R., Long, D., Fox, M.: VAL: automatic plan validation, continuous effects and mixed initiative planning using PDDL. In: 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI). pp. 294–301 (2004)
11. Jimenez, P.: Survey on model-based manipulation planning of deformable objects. Robotics and Computer-Integrated Manufacturing 28(2), 154–163 (2012)
12. Knapper, R., Layton, T., Romanishin, J., Rus, D.: Ikeabot: an autonomous multi-robot coordinated furniture assembly system. In: Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA 2013). Karlsruhe, Germany (May 2013)
13. Krotzsch, M., Simancik, F., Horrocks, I.: A description logic primer. arXiv:1201.4089v3 (2013)
14. Lemaignan, S., Ros, R., Mösenlechner, L., Alami, R., Beetz, M.: Oro, a knowledge management platform for cognitive architectures in robotics. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 3548–3553. IEEE (2010)
15. Lipovetzky, N., Ramirez, M., Muise, C., Geffner, H.: Width and inference based planners: SIW, BFS(f), and PROBE. Proceedings of the 8th International Planning Competition (IPC-2014) (2014)
16. McDermott, D.: The 1998 AI planning systems competition. AI Magazine 21(2), 35–55 (2000)
17. Newman, W., Chong, Z.H., Du, C., Hung, R., Lee, K.H., Ma, L., Ng, T., Swetenham, C., Tjoeng, K., Wang, W.: Autonomous valve turning with an Atlas humanoid robot. In: Proceedings

of the 2014 IEEE-RAS International Conference on Humanoid Robotics (HUMANOIDS 2014). Madrid, Spain (November 2014)

18. Rintanen, J.: Madagascar: Scalable planning with SAT. In: Proceedings of the 8th International Planning Competition (IPC-2014) (2014)
19. 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, Lausanne, Switzerland (2016)
20. Smith, C., Karayiannidis, Y., Nalpantidis, L., Gratal, X., Qi, P., Dimarogonas, D., Kragic, D.: Dual arm manipulation: a survey. Robotics and Autonomous Systems 60(10), 1340–1353 (2012)
21. Tenorth, M., Beetz, M.: Representations for robot knowledge in the knowrob framework. Artificial Intelligence 247, 151–169 (2017)
22. Vallati, M., Chrpa, L., Grzes, M., McCluskey, T., Roberts, M., Sanner, S.: The 2014 international planning competition: Progress and trends. AI Magazine (2015)
23. Vidal, V.: Yahsp3 and yahsp3-mt in the 8th international planning competition. In: Proceedings of the 8th International Planning Competition (IPC-2014) (2014)
24. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin 1(6), 80–83 (1945)
25. 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)