Declarative Solutions for the the Manipulation of Articulated Objects Using Dual-Arm Robots

Riccardo Bertolucci
richi.bertolucci@gmail.com

DIBRIS, University of Genova, Italy

Abstract. The manipulation of flexible object is of primary importance in industry 4.0 and in home environments scenarios. Traditionally, this problem has been tackled by developing ad-hoc approaches, that lack of flexibility and portability.
We propose an approach in which a flexible object is modelled as an articulated object, or rather, a set of links connect via joints In this paper we present an extended analysis of the framework based on Answer Set Programming (ASP) for the automated manipulation of articulated objects in a robot architecture. In detail, we modeled the same scenario with different grades of precision: a simple model it is used to describe the scenario with an high level of abstraction, while an extended model it used to include more detail and therefore to increase the represented knowledge of such scenario. With respect to the simple reference scenario we analyse the behaviours of our strategy for the action planning module, while, for the extended scenario we introduce the concept of macro action and we then we analyse their performances w.r.t our problem.
Our aim is to have an understanding of the performances of these approaches with respect to planning time and execution time as well.[1]

Keywords: Answer Set Programming, Robots Manipulation, Macro actions

1 Introduction

The manipulation of articulated objects is of primary importance in robotics, and is one of the most complex robotics tasks [1,2]. Traditionally, this problem has been tackled by developing ad-hoc approaches, that lack of flexibility and portability. The development of new software, algorithm and strategies, together with the improvements in the mechanical design for grippers and robotic hands, for autonomous robots with robust manipulation skills, can lead to breakthroughs in various applications, such as humanoid robots, horticulture harvesting grasping, human robot interaction, planetary exploration, flexible manufacturing and much more. This gives the possibility of addressing some of the issues related to robotized work, such as mechanical design issues, control issues, modelling achievements and issues, applications in industrial field and non-conventional applications (including, for example, service robotics and agriculture)[3,1]. In the past years attention has been paid to the development of

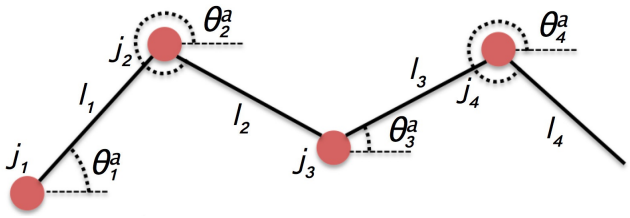

Fig. 1: The articulated object representation.

approaches and algorithms for generating the sequence of movements a robot has to perform in order to manipulate an articulated object but the issue is still not being fully addressed. In the literature, the problem of determining the two-dimensional (2D) configuration of articulated or flexible objects has received much attention in the past few years [4,5,6,7], whereas the problem of obtaining a target configuration via manipulation has been explored in motion planning [8,9,10]. A limitation of such manipulation strategies is that they 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, in these contexts generalisation property and scalability are somehow limited.

In this paper we present the analysis of a framework based on Answer Set Programming (ASP) [11,12,13,14,15,16,17,18] for the automated manipulation of articulated objects in a robot 2D work-space. ASP is a general, prominent knowledge representation and reasoning language with roots in logic programming and non-monotonic reasoning [19,20], with readable syntax and clear semantics [21]. In particular, ASP is employed for representing the configuration of the articulated object, for checking the consistency of the knowledge base, as well as for generating the sequence of manipulation actions, i.e. the plan.

2 Problem Statement and the Simple Reference Scenario

In this section we define the problem addressed, and we present the considered first reference scenario.

2.1 Problem Statement

Our goal is to present (i) an overview of the ASP-based architecture for the manipulation of articulated objects in terms of the representation of the desired scenario and the planning strategies developed for the selection of manipulation actions aimed to maximises the reliability of the robot execution, and (ii) give a simple overview on how we modeled macro actions inside our ASP-based architecture.

An articulated object is defined as a pair $\alpha = (\mathcal{L}, \mathcal{J})$, where $\mathcal{L}$ is the ordered set of its $|\mathcal{L}|$ links and $\mathcal{J}$ is the ordered set of its $|\mathcal{J}|$ joints. Each link $l \in \mathcal{L}$ is characterised by two parameters, namely a length λ_l and an orientation θ_l. We allow only for a limited

number of possible orientations, which induces a finite set of allowed angle orientations for each link. The configuration of an articulated object is modeled as $|L|$-ple:

$$C_{\alpha,a} = \left(\theta^a_1, \ldots, \theta^a_{|L|}\right). \tag{1}$$

2.2 Modeled Scenarios

Simple Model We relay on QR codes to compute and provide to the architecture an overall link pose, which directly maps to an absolute link orientation θ^a_l.

Extended model This scenario it is developed in order to describe with an higher accuracy the robot and its work-space. However, this model does not modify any physical characteristics with respect to the setup introduced in the previous paragraph. Herewith we briefly describe such modelling, and whenever relevant we highlight the main modifications we introduced to the initial scenario. Firstly, *The robot grippers are now explicitly modelled*. Each gripper is now considered as a resource that can be *occupied* (i.e., keeping a link firmly, or rotating a link) or *free*. This open the possibility of representing which gripper will manipulate a given link. Then, *each time a manipulation action is carried out on a given link, it is assured that the link is centred in the robot workspace*. This is due to the fact that, due to the physical property of the object, some of the link can be positioned outside the reach of the robot arms. Finally, *grasping and release actions by the two grippers are now explicitly modelled.*

All the above mentioned features allow for mainly two improvements: on the one hand, the encoding is expected to be able to better manage the explicitly modelled robot resources (i.e., the grippers); on the other hand, manipulation actions are characterised by a more precise semantics, which does not make any implicit assumption about actual robot behaviour.

3 The Robot Architecture

The architecture of the Baxter from Rethink Robotics is shown in Figure 2. In the current implementation, perception is managed using a camera sensor located on top of the robot's *head* and pointing downward, which provides 6D poses for each link, and in turn update corresponding ASP-based representation structures in the *Knowledge Base* module. The *Consistency Checking* module performs a check for knowledge base validation. In case the check succeeds, the *Goal Checker* module is notified and it process the information given by the *Knowledge Base* in order to compute which requirements are already fulfilled. In fact, usually due to human intervention, some constrains included in the goal can be already achieved. Tacking this in consideration the problem is then generated. The *Action Planner* module receives such problem instance and generates a plan in the form of a suitable sequence of actions to be performed. Once a plan is generated, its actions are processed sequentially to drive the overall behaviour of the robot *Motion Planner* module, which is responsible of the execution.

For the sake of brevity we will focus only on the *Action Planning* module and how we modified it in order to implement different approaches.

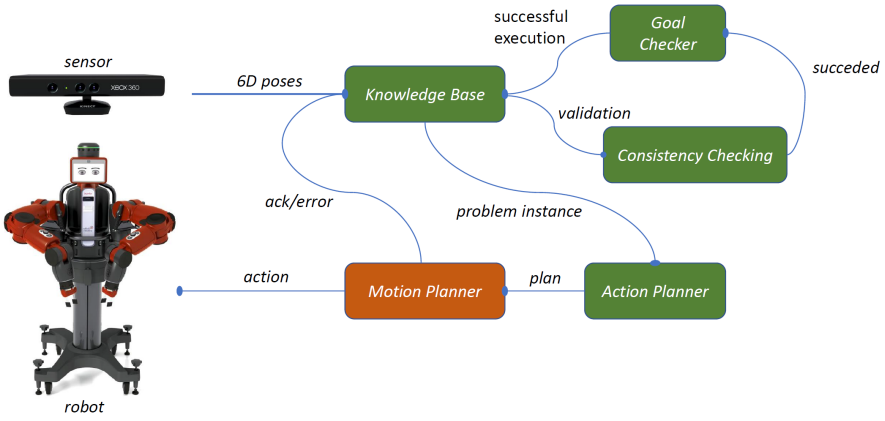

Fig. 2: The robot's architecture: in *green* the ASP-based modules, in *orange* the ROSPlan-based module.

4 Action Planning Module

In this section we describe how ASP is used to implement the Action Planning Module depicted. In the following, we assume the reader is familiar with ASP and ASP-Core-2 input language specification [22].

ASP is not a planning-specific language, but it can be also used to specify encoding for planning domains [23], like our target problem. We have defined several encoding variants, for what concerns either the manipulation modes and the strategy for computing plans.

4.1 Simple Model

The encoding described in this section is embedded into a classical iterative deepening approach in the spirit of SAT-based planning [24], in which the maximum number of step allowed, called `timemax`, is initially set to 1 and then increased by 1 if a plan is not found. This guarantees the computation of an optimal plan, w.r.t. the number of action to perform, that is the shortest possible plans for a sequential encoding, i.e., when the robot performs only one action for each step.

4.2 Extended Model

Inside the context of the extended scenario (see Section 2.2), we explored two different strategies in order to investigate the pros and drawbacks of each approach. Moreover, in this scenario, we investigate the propriety of the macros. The two strategies are:

– Simple Actions Extended Scenario (SAES): This encoding models the same actions as the Standard Strategy in the simple scenario but it includes also the robot resources that can be occupied at each time step (i.e. the robot gripper);
– Macro Actions Extended Scenario (MAES): Here we have modelled the same scenario as in the SAES. The difference consists of the modelled actions: sets of simple actions are gathered inside just one atoms.

Representing Macros in ASP Given a rule r_i representing an action, *pre*(r_i) denotes the body of the rule. Intuitively, it represents the conditions that must hold in order to activate the action represented by the rule. Moreover, *del*(r_i) (resp. *add*(r_i)) represent all the atoms that are set as false (resp. true) whenever the conditions denoted by *pre*(r_i) hold. We encoded a macro action as a single choice rules composed by a fresh atom in its head containing all the variables appearing in its body. Furthermore, the choice rule body it is composed as follows: a macro $r_{i,j}$ is constructed by assembling the rules representing single actions and by generating *pre*$(r_{i,j})$, *del*$(r_{i,j})$, and *add*$(r_{i,j})$, as follows:

– *pre*$(r_{i,j}) =$ *pre*$(r_i) \cup$ (*pre*$(r_j) \setminus$ *add*$(r_i))$
– *del*$(r_{i,j}) =$ (*del*$(r_i) \setminus$ *add*$(r_j)) \cup$ *del*(r_j)
– *add*$(r_{i,j}) =$ (*add*$(r_i) \setminus$ *del*$(r_j)) \cup$ *add*(r_j)

where r_i and r_j are two distinct rules. Then, for a macro $r_{i,j}$, the body of the choice rule is represented by *pre*$(r_{i,j})$. The macro composed as such represent multiple simple action and their effect that are modeled as a set of several simple rules.

Macros for the Extended Scenario. The following macros have been considered:

– `linkToCentral_take`: it is the composition of two actions $move\_link\_to\_central$, that moves the articulated object so that the joint in between the links that have to be manipulated is in the centre of the workspace, and $takes\_links\_to\_move$, that grasps the links to be manipulated. As links cannot be grasped by the robot if they are not in the centre of the workspace, this macro aims at providing a single rule for cases where links are not in the right position.
– `changeAngle_release`: it is the composition of the choice rule $changeAngle$, that changes the angle of a link, and $release\_links$, that releases the links currently grasped. This macro aims at providing a single rule for cases where it is necessary to act on a link and then releasing it.
– `take_changeAngle_release`: represents the composition of $takes\_links\_to\_move$, $changeAngle$, and $release\_links$. This macro aims at providing a single action for cases where it is necessary to act on a link that was already in the center of the workspace.

.

5 Results and Conclusions

In order to obtain an overview of the capabilities of the considered encodings we used as test problems the same problems, with the due adaptations for each model. Eventually,

we had 320 instances with 4 and 6, and granularity values of 4, 6, 8 or 12 possible angles, 10 instances for each pair (number of links, granularity). For each testing instance time limit of 300 seconds and memory limit of 16 GB was applied. Clingo was used to solve the ASP-encoded instances. All the experiments were conducted on Intel i7-4790 CPU and Linux OS. We compared the performance of the considered encodings using coverage (percentage of solved instances) and PAR10. Penalised Average Runtime (PAR10) score is a metric usually exploited in machine learning and algorithm configuration techniques. This metric trades off coverage and runtime for solved problems: if an encoding e allows the solver to solve an instance Π in time $t \leq T$ ($T = 300$s in our case), then $PAR10(e, \Pi) = t$, otherwise $PAR10(e, \Pi) = 10 \times T$ (i.e., 3000s in our case). The above tables summarises the results achieved by Clingo for solving in-

Number Angles: 4

PAR10

	Standard	SAES	MAES
4	0.0	3.82	0.66
5	0.04	929.23	9.9
6	2.8	2104.76	665.19
7	10.38	3000.0	1265.75
8	1235.98	3000.0	2448.22
10	1339.36	3000.0	3000.0
11	2105.99	3000.0	3000.0
12	2704.99	3000.0	3000.0

Coverage

	Standard	SAES	MAES
4	100	100	100
5	100	70	100
6	100	30	80
7	100	0	60
8	60	0	20
10	50	0	0
11	30	0	0
12	10	0	0

Number Angles: 6

PAR10

	Standard	SAES	MAES
4	0.08	16.36	4.43
5	1.52	2402.65	427.06
6	636.69	3000.0	2701.3
7	3000.0	3000.0	2439.51
8	2403.78	3000.0	3000.0
10	2697.74	3000.0	3000.0
11	2719.29	3000.0	3000.0
12	3000.0	3000.0	3000.0

Coverage

	Standard	SAES	MAES
4	100	100	100
5	100	20	90
6	80	0	10
7	0	0	20
8	20	0	0
10	10	0	0
11	10	0	0
12	0	0	0

stances encoded in the *Standard Strategy*, *SAES*, and *MAES*. It is worth reminding that the *Standard Strategy* encoding is much more simplistic then the others, as it ignores the position of the links to be manipulated, and considers high level actions that have to be broken down into a large number of low-level primitives. On the contrary, *SAES* and *MAES* encodings provide a more detailed and rich description of the problem, that allows to generate plans that are easier to be put in place by the manipulator. So, a direct comparison between the *Standard Strategy* and *SAES/MAES* is not possible, but it is nonetheless interesting to have also the results obtained by the *Standard Strategy*. The comparison of the performance achieved by Clingo when using the *SAES* and *MAES* encodings can shed some light on the usefulness of the macros. It is easy to notice that the use of macros allows Clingo to solve a larger number of instances, and that macros are generally helpful in improving the runtime.

In the future we would like to test with other ASP solvers, e.g. WASP [25], and to inject in these solvers heuristics and algorithms, e.g. [26,27,28], that proved to be effective in the planning domain.

References

1. J. Krüger, T. K. Lien, and A. Verl, "Cooperation of human and machines in assembly lines," *CIRP Annals*, vol. 58, no. 2, pp. 628 – 646, 2009.
2. C. Heyer, "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.
3. C. Heyer, "Human-robot interaction and future industrial robotics applications," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4749–4754, IEEE, 2010.
4. A. Capitanelli, M. Maratea, F. Mastrogiovanni, and M. Vallati, "Automated planning techniques for robot manipulation tasks involving articulated objects," in *Proceedings of the International Conference of the Italian Association for Artificial Intelligence (AI*IA)*, pp. 483–497, Springer, 2017.
5. A. Capitanelli, M. Maratea, F. Mastrogiovanni, and M. Vallati, "On the manipulation of articulated objects in human-robot cooperation scenarios," *Robotics and Autonomous Systems*, vol. 109, pp. 139–155, 2018.
6. A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2146–2153, IEEE, 2017.
7. H. Wakamatsu, E. Arai, and S. Hirai, "Knotting/unknotting manipulation of deformable linear objects," *International Journal of Robotic Research*, vol. 25, no. 4, pp. 371–395, 2006.
8. L. Bodenhagen, A. R. Fugl, A. Jordt, M. Willatzen, K. A. Andersen, M. M. Olsen, R. Koch, H. G. Petersen, and N. Krüger, "An adaptable robot vision system performing manipulation actions with flexible objects," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 3, pp. 749–765, 2014.
9. J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from demonstrations through the use of non-rigid registration," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, pp. 339–354, Springer, 2013.
10. Y. Yamakawa, A. Namiki, and M. Ishikawa, "Dynamic high-speed knotting of a rope by a manipulator," *IJARS*, vol. 10, pp. 1–12, 2013.
11. M. Gebser, M. Maratea, and F. Ricca, "The Design of the Sixth Answer Set Programming Competition," in *LPNMR*, vol. 9345 of *LNCS*, pp. 531–544, Springer, 2015.
12. M. Gebser, M. Maratea, and F. Ricca, "What's hot in the answer set programming competition," in *AAAI*, pp. 4327–4329, AAAI Press, 2016.
13. F. Calimeri, M. Gebser, M. Maratea, and F. Ricca, "Design and results of the Fifth Answer Set Programming Competition," *Artificial Intelligence*, vol. 231, pp. 151–181, 2016.
14. M. Gebser, M. Maratea, and F. Ricca, "The sixth answer set programming competition," *Journal of Artificial Intelligence Research*, vol. 60, pp. 41–95, 2017.
15. M. Gebser, M. Maratea, and F. Ricca, "The design of the seventh answer set programming competition," in *Proc. of the 14th International Conference Logic Programming and Nonmonotonic Reasoning, LPNMR 2017* (M. Balduccini and T. Janhunen, eds.), vol. 10377 of *Lecture Notes in Computer Science*, pp. 3–9, Springer, 2017.

16. Y. Lierler, M. Maratea, and F. Ricca, "Systems, engineering environments, and competitions," *AI Magazine*, vol. 37, no. 3, pp. 45–52, 2016.
17. M. Gebser, N. Leone, M. Maratea, S. Perri, F. Ricca, and T. Schaub, "Evaluation techniques and systems for answer set programming: a survey," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018* (J. Lang, ed.), pp. 5450–5456, ijcai.org, 2018.
18. M. Gebser, M. Maratea, and F. Ricca, "The seventh answer set programming competition: Design and results," *Theory and Practice of Logic Programming*, 2020. To appear.
19. M. Gelfond and V. Lifschitz, "The stable model semantics for logic programming," in *Proceedings of the International Conference on Logic Programming (ICLP)*, pp. 1070–1080, MIT Press, 1988.
20. M. Gelfond and V. Lifschitz, "Classical negation in logic programs and disjunctive databases," *New Generation Computing*, vol. 9, pp. 365–385, 1991.
21. F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, M. Maratea, F. Ricca, and T. Schaub, "Asp-core-2 input language format," *Theory and Practice of Logic Programming*, 2020. To appear.
22. F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, F. Ricca, and T. Schaub, "ASP-Core-2 Input Language Format," 2013.
23. V. Lifschitz, "Answer set programming and plan generation," *Artificial Intelligence Journal*, vol. 138, no. 1-2, pp. 39–54, 2002.
24. H. A. Kautz and B. Selman, "Planning as satisfiability," in *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pp. 359–363, 1992.
25. M. Alviano, G. Amendola, C. Dodaro, N. Leone, M. Maratea, and F. Ricca, "Evaluation of disjunctive programs in WASP," in *Proc. of the 15th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2019* (M. Balduccini, Y. Lierler, and S. Woltran, eds.), vol. 11481 of *Lecture Notes in Computer Science*, pp. 241–255, Springer, 2019.
26. E. Giunchiglia, M. Maratea, and A. Tacchella, "Dependent and independent variables in propositional satisfiability," in *Proceedings of the European Conference on Logics in Artificial Intelligence*, vol. 2424 of *Lecture Notes in Computer Science*, pp. 296–307, Springer, 2002.
27. E. Giunchiglia, M. Maratea, and A. Tacchella, "(In)effectiveness of look-ahead techniques in a modern SAT solver," in *Proceedings of 9th International Conference on the Principles and Practice of Constraint Programming*, vol. 2833 of *Lecture Notes in Computer Science*, pp. 842–846, Springer, 2003.
28. E. DiRosa, E. Giunchiglia, and M. Maratea, "A new approach for solving satisfiability problems with qualitative preferences," in *Proc. of the 18th European Conference on Artificial Intelligence, Patras, ECAI 2008* (M. Ghallab, C. D. Spyropoulos, N. Fakotakis, and N. M. Avouris, eds.), vol. 178 of *Frontiers in Artificial Intelligence and Applications*, pp. 510–514, IOS Press, 2008.