KR&R Approaches for Robot Manipulation Tasks with Articulated Objects

Riccardo Bertolucci[1], Alessio Capitanelli[2], Carmine Dodaro[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. In this paper we present two approaches for solving robot manipulation tasks with articulated objects by using knowledge representation and reasoning languages and tools. Such languages and tools are used both for representing initial and final configurations from an ontology description and for planning the robot (manipulation) actions. In the first approach, standard PDDL language and solvers are used to plan those actions, and DL solvers for ontology consistency checking. In the second (ongoing) approach, ASP is employed as a unifying framework for both ontology checking and planning.

1 Introduction

Articulated objects are made up of links connected via joints that can move with respect to each other. The manipulation of such objects (which can be considered as a good approximation for strings, ropes or cables) is of the utmost importance in different application scenarios [25,35].

Apart from robot manipulation actions, the configurations of such objects depend on their parts and may be the result also of external factors, such as the constraints imposed by the geometry of the environment or the effects of gravity. This leads to a multi-faceted representation problem: on the one hand, we must address how to maintain the representation of articulated (or flexible) objects depending on how they are perceived by the robot; on the other hand, we must ground reasoning on such representation to manipulate such objects in order to obtain a given goal configuration.

In the literature, a number of *ad hoc* solutions have been discussed, including ones where robots manipulate ropes [37], cables [11], tie or untie knots [36], or operate on mobile parts of their environment, e.g., various handles, furniture or valves [16,27], even in human-robot collaborative scenarios [28]. However, these approaches are characterized by at least one of two assumptions: the first posits that manipulation actions are directly based on perceptual data and, therefore, on the specific geometrical problem at hand [10,11,33], whereas the second one

argues that an *a priori* physical model of the object to manipulate is either known or learned [26,32].

In this paper we present two action planning and execution architectures for robot manipulation tasks with articulated objects. The two architectures are aimed at reasoning about any pair of abstract representations of articulated or flexible objects and the transitions induced by an appropriate sequence of manipulation actions, based on knowledge representation and reasoning (KR&R) languages and tools. Such languages and tools are used both for representing initial and final object configurations in an ontology-based description and for planning the robot manipulation actions. Both architectures are under testing on a robot hybrid reactive/deliberative framework using a dual-arm Baxter robot from Rethink Robotics.

In the first architecture (outlined in Section 2), the standard Planning Domain Description Language (PDDL) and domain-independent solvers are used for modeling and planning those actions, and DL solvers are employed to check for consistency in an OWL ontology [14]. In the second architecture, which is currently under validation (see Section 3), Answer Set Programming (ASP) is employed as a unifying framework for both planning and ontology checking. The two architectures are compared (see Section 4) in terms for action planning performance. In Section 5 we highlight possible future research directions.

2 Architecture #1

Capitanelli et al. [14] introduced a hybrid reactive/deliberative architecture for robots based on PDDL and OWL2. The architecture, which extends the well-known ROSPlan architecture [15], adopts the ARMOR framework[4], as well as two state-of-the-art planners, namely Probe [30] and Madagascar [34], along with the MoveIt![5] motion planning library. Generated plans are validated by the VAL plan validator.

Two different manipulation modes are allowed, and therefore we consider data obtained by each of the two following modes:

- CAPF: for any given link, it is possible to operate only on the successive link, and therefore only forward motion propagation is allowed, e.g., it is possible to move *link 2* only keeping *link 1* firmly and acting on *link 2*;
- CAPFD: it is possible to move each link in each direction, therefore either forward or back propagation is allowed, e.g., it is possible to move *link 2* by grasping *link 1* or *link 3* and then operating on *link 2*.

As far as the ontology is concerned, the architecture is able to store and compute the differences between initial (and, in general, current) and goal object configurations, in terms of normative knowledge. In this way, we can remove unused constraints from the problem file and thus alleviate the planner's workload. The architecture computes these differences at each action execution step.

[4] Web: `https://github.com/EMAROlab/armor`
[5] Web: `http://moveit.ros.org/`

Due this this feature, we can use SWRL rules to enforce plan execution robustness and flexibility: if a given link is accidentally misplaced by the robot, or a human interacting with the robot does it purposely during plan execution, SWRL rules compute the difference between current and goal configurations and bootstrap a new planning process.

3 Architecture #2

The second architecture, which is subject of on-going work, is similar to the first one, but it is completely based on ASP. An architecture based on a unified logic framework is expected to have better performance or to find solutions to the planning problem that are better optimized with respect to different parameters. The parameter we aim at optimizing (in this case, minimizing) is the number of actions computed by the planner.

Differently from PDDL, ASP is a general purpose language for a variety of applications (e.g.[8,6,7]) and not devoted to automated planning, but it can be used for planning purposes as well given the efficiency of ASP solvers, witnessed by the results of a number of ASP-related competitions, e.g.[22,23,13,24,29,21]. Moreover, ASP can deal with ontology management.

We want to compute plans with both Clingo [18], i.e., the combination of the grounder Gringo [19] and the solver Clasp [20], and DLV2 [1], i.e., the combination of the grounder I-DLV [12] and the solver WASP [4], to have an overview of the performance of ASP-based planners [31].

ASP solutions to the planning problem are translated to the same output format used by the standard PDDL-based planners, which is to be checked by VAL.

We tried different options for the ASP-based planner:

- *Wrapper*. We compute the plan varying the number of allowed maximum steps starting from 1 and increasing it by one unit if the plan is not found. This allows us to find the optimal solution in terms of the number of actions but sacrificing CPU performance.
- *Weak Constraint* [2,3,5]. We add to the domain a weak constraint on the maximum number of allowed steps.
- *Random*. We select the maximum number of steps randomly. This does not ensure an optimal solution neither with respect to the number of actions nor in terms of planning computation time.

Currently, we are completing the development and the integration of the ASP-based ontology component by means of an ASP encoding.

4 Results

In this section we discuss preliminary results obtained for the planning module in the second architecture. As a reference, we analyze two of the experiments we

carried out. For each experiment, we have two different tables: the first shows the planning execution time in seconds, the second shows the number of actions as computed by the planners. Each table contains results for different set-ups:

- Madagascar: results obtained using the PDDL solver Madagascar [34].
- Probe: results obtained with the PDDL solver Probe [30].
- Clingo: results obtained with the ASP solver Clingo using the *Wrapper* set-up as discussed above.
- Clingo Weak: results obtained with the ASP solver Clingo using the *Weak Constraint* set-up as explained before.
- Clingo Not Optimal: results obtained with the ASP solver Clingo using the *Random* set-up as discussed above.

The name of each experiment is composed by two numbers, respectively representing the number of joints of the articulated object and the number of possible angles that a link can assume. For each experiment, 10 different problem instances, with different initial states and goals, were tested, with a timeout of 1 hour. The median value is computed and shown in the tables. A "TIME" (resp. -1) indicates that the solver can not find a solution (resp. a plan) within an hour.

4.1 Experiment 5_6 (CAPF)

Experiment Number	1	2	3	4	5	6	7	8	9	10
Madagascar	0.15	0.31	0.1	0.18	0.02	0.18	0.03	0.1	0.11	0.13
Probe	0.01	0.05	0.02	0.02	0.01	0.01	0.02	0.01	0.01	0.01
Clingo	1.53	4.81	1.88	1.91	0.01	2.20	0.01	0.01	1.46	1.71
Clingo Weak	6.26	11.86	6.77	6.24	0.32	6.76	0.41	1.42	5.23	5.00
Clingo Not Optimal	2.02	2.45	0.97	3.48	0.32	4.53	0.88	5.17	9.25	2.45

Table 1: CPU times.

In tables 1 and 2 we show an experiment with a medium size problem (5 links and 6 possible angles). We have different result depending on the selected approach:

- Clingo: As we said we have always the optimal solution. This result comes at the cost of a longer execution time to compute the plan.
- Clingo Weak: As before it always finds an optimal solution. With this approach we avoid the need of external script (e.g. wrapper), but it comes with a slower execution time to compute the plan.

Experiment Number	1	2	3	4	5	6	7	8	9	10
Madagascar	13	13	9	13	4	13	4	7	13	9
Probe	9	11	9	11	4	11	4	13	9	9
Clingo	9	9	9	9	4	9	4	7	9	9
Clingo Weak	9	9	9	9	4	9	4	7	9	9
Clingo Not Optimal	43	45	39	45	4	45	40	39	49	13

Table 2: Number of computed actions.

- Clingo Not Optimal: the computed solutions are way larger than the ones computed by the two previous approaches. We expected a not optimal solution but we also expected to have a faster execution time. however, as the table shows, the execution times is similar to the times of the Clingo Weak approach.

4.2 Experiment 7_6 (CAPF)

Experiment Number	1	2	3	4	5	6	7	8	9	10
Madagascar	0.38	0.26	0.8	0.48	0.05	0.61	0.13	0.08	0.28	0.65
Probe	0.13	0.03	0.05	0.07	0.07	0.05	0.27	0.08	0.25	0.09
Clingo	1.77	2.43	TIME	184.71	0.01	TIME	2410.96	1.56	1006.86	19.13
Clingo Weak	7.85	8.86	TIME	614.90	0.91	TIME	2116.66	4.85	1119.21	24.70
Clingo Not Optimal	10.27	28.52	25.51	23.50	5.1	66.20	4.13	22.48	37.64	7.72

Table 3: CPU times.

Experiment Number	1	2	3	4	5	6	7	8	9	10
Madagascar	14	10	33	22	5	18	14	8	12	21
Probe	10	8	15	12	5	14	12	8	20	11
Clingo	8	8	-1	12	5	-1	12	8	12	9
Clingo Weak	8	8	-1	12	5	-1	12	8	12	9
Clingo Not Optimal	18	48	15	48	21	42	48	46	46	47

Table 4: Number of computed actions.

In tables 3 and 4 we show an experiment with a larger problem (7 links and 6 possible angles). We have different result depending on the selected approach:

- Clingo: As for the previous problem, the solution is always optimal and this, in some cases, leads us to plans that are 50% smaller than the ones computed from the PDDL solvers. However, since the problem is bigger than before, some plans require too much time to be computed by the ASP solvers and consequently we are unable to have a solutions for those problems.
- Clingo Weak: As before we ensure an optimal solution but we encounter the same problems we have with the Clingo approach.
- Clingo Not Optimal: the computed solutions are way larger than the ones computed by the two previous approaches. However it can be noticed from the table that, even though the execution time is high, the plan is always computed within the time limit.

The ASP-based approach is able to return plans sometimes significantly smaller than in the previous solution, sometimes at the price of increased CPU time. However, we should take into account that in real environment action's execution is not instantaneous (hypothesis of classical planning), but takes time, so the total time for executing the plan by the architecture could be highly influenced by the number of performed actions.

5 Conclusions and future work

In this paper, we have presented two KR&R approaches for solving robot manipulation tasks with articulated objects.

Current and future work include:

- Completing the ASP-based framework: the implementation of the storage module has to be finished and validated.
- Testing the architecture, in particular the planning module, with DLV2.
- Testing the architectures on different robot platforms to evaluate the portability of our solutions, and adding more KR&R approaches, e.g. using the mixed discrete-continuous approach of PDDL+ [17] and CASP [9].

References

1. Alviano, M., Calimeri, F., Dodaro, C., Fuscà, D., Leone, N., Perri, S., Ricca, F., Veltri, P., Zangari, J.: The ASP system DLV2. In: LPNMR. Lecture Notes in Computer Science, vol. 10377, pp. 215–221. Springer (2017)
2. Alviano, M., Dodaro, C.: Anytime answer set optimization via unsatisfiable core shrinking. TPLP 16(5-6), 533–551 (2016)
3. Alviano, M., Dodaro, C.: Unsatisfiable core shrinking for anytime answer set optimization. In: IJCAI. pp. 4781–4785. ijcai.org (2017)
4. Alviano, M., Dodaro, C., Leone, N., Ricca, F.: Advances in WASP. In: LPNMR. Lecture Notes in Computer Science, vol. 9345, pp. 40–54. Springer (2015)

5. Alviano, M., Dodaro, C., Marques-Silva, J., Ricca, F.: Optimum stable model search: Algorithms and implementation. J. Log. Comput. http://dx.doi.org/10.1093/logcom/exv061, in press
6. Amendola, G., Dodaro, C., Leone, N., Ricca, F.: On the application of answer set programming to the conference paper assignment problem. In: AI*IA. Lecture Notes in Computer Science, vol. 10037, pp. 164–178. Springer (2016)
7. Amendola, G., Greco, G., Leone, N., Veltri, P.: Modeling and reasoning about NTU games via answer set programming. In: IJCAI 2016. pp. 38–45 (2016)
8. Balduccini, M., Gelfond, M., Watson, R., Nogueira, M.: The USA-Advisor: A case study in answer set planning. In: LPNMR. LNCS, vol. 2173, pp. 439–442. Springer (2001)
9. Balduccini, M., Magazzeni, D., Maratea, M., LeBlanc, E.: Casp solutions for planning in hybrid domains. arXiv preprint arXiv:1704.03574 (2017)
10. Berenson, D.: Manipulation of deformable objects without modeling and simulating deformation. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on. pp. 4525–4532. IEEE (2013)
11. Bodenhagen, L., Fugl, A.R., Jordt, A., Willatzen, M., Andersen, K.A., Olsen, M.M., Koch, R., Petersen, H.G., Krüger, N.: An adaptable robot vision system performing manipulation actions with flexible objects. IEEE transactions on automation science and engineering 11(3), 749–765 (2014)
12. Calimeri, F., Fuscà, D., Perri, S., Zangari, J.: I-DLV: the new intelligent grounder of DLV. Intelligenza Artificiale 11(1), 5–20 (2017)
13. Calimeri, F., Gebser, M., Maratea, M., Ricca, F.: Design and results of the Fifth Answer Set Programming Competition. Artif. Intell. 231, 151–181 (2016)
14. 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)
15. 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: ICAPS. pp. 333–341 (2015)
16. Dang, H., Allen, P.K.: Robot learning of everyday object manipulations via human demonstration. In: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on. pp. 1284–1289. IEEE (2010)
17. Fox, M., Long, D.: Modelling mixed discrete-continuous domains for planning. J. Artif. Intell. Res. 27, 235–297 (2006)
18. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Wanko, P.: Theory solving made easy with clingo 5. In: ICLP (Technical Communications). OASICS, vol. 52, pp. 2:1–2:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)
19. Gebser, M., Kaminski, R., König, A., Schaub, T.: Advances in *gringo* series 3. In: LPNMR. Lecture Notes in Computer Science, vol. 6645, pp. 345–351. Springer (2011)
20. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artif. Intell. 187, 52–89 (2012)
21. Gebser, M., Leone, N., Maratea, M., Perri, S., Ricca, F., Schaub, T.: Evaluation techniques and systems for answer set programming: a survey. In: Lang, J. (ed.) Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018. pp. 5450–5456. ijcai.org (2018)
22. Gebser, M., Maratea, M., Ricca, F.: The Design of the Sixth Answer Set Programming Competition. In: LPNMR. LNCS, vol. 9345, pp. 531–544. Springer (2015)

23. Gebser, M., Maratea, M., Ricca, F.: What's hot in the answer set programming competition. In: AAAI. pp. 4327–4329. AAAI Press (2016)
24. Gebser, M., Maratea, M., Ricca, F.: The sixth answer set programming competition. Journal of Artificial Intelligence Research 60, 41–95 (2017)
25. Henrich, D., Wörn, H.: Robot manipulation of deformable objects. Springer Science & Business Media (2012)
26. Howard, A.M., Bekey, G.A.: Recursive learning for deformable object manipulation. In: Advanced Robotics, 1997. ICAR'97. Proceedings., 8th International Conference on. pp. 939–944. IEEE (1997)
27. Knepper, R.A., Layton, T., Romanishin, J., Rus, D.: Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In: Robotics and Automation (ICRA), 2013 IEEE International Conference on. pp. 855–862. IEEE (2013)
28. Kruse, D., Radke, R.J., Wen, J.T.: Collaborative human-robot manipulation of highly deformable materials. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on. pp. 3782–3787. IEEE (2015)
29. Lierler, Y., Maratea, M., Ricca, F.: Systems, engineering environments, and competitions. AI Magazine 37(3), 45–52 (2016)
30. Lipovetzky, N., Geffner, H.: Searching for plans with carefully designed probes. In: ICAPS. pp. 154–161 (2011)
31. M. Gebser, M. Maratea, F.R.: The sixth answer set programming competition. j. artif. intell. res. 60: 41-95 (201)
32. Meier, U., López, O., Monserrat, C., Juan, M.C., Alcaniz, M.: Real-time deformable models for surgery simulation: a survey. Computer methods and programs in biomedicine 77(3), 183–197 (2005)
33. Miller, S., Van Den Berg, J., Fritz, M., Darrell, T., Goldberg, K., Abbeel, P.: A geometric approach to robotic laundry folding. The International Journal of Robotics Research 31(2), 249–267 (2012)
34. Rintanen, J.: Madagascar: Scalable planning with sat. Proceedings of the 8th International Planning Competition (IPC-2014) (2014)
35. Saadat, M., Nan, P.: Industrial applications of automatic manipulation of flexible materials. Industrial Robot: An International Journal 29(5), 434–442 (2002)
36. Wakamatsu, H., Arai, E., Hirai, S.: Knotting/unknotting manipulation of deformable linear objects. The International Journal of Robotics Research 25(4), 371–395 (2006)
37. Yamakawa, Y., Namiki, A., Ishikawa, M.: Dynamic high-speed knotting of a rope by a manipulator. International Journal of Advanced Robotic Systems 10(10), 361 (2013)