Towards Integration of Multi-Agent Planning with Self-Organising Collective Processes

Towards Integration of Multi-Agent Planning with Self-Organising Collective Processes

Giorgio Audrito
Dipartimento di Informatica
Università di Torino
Torino, Italy
0000-0002-2319-0375

Roberto Casadei
Department of Computer Science and Engineering
Università di Bologna
Bologna, Italy
0000-0001-9149-949X

Gianluca Torta
Dipartimento di Informatica
Università di Torino
Torino, Italy
0000-0002-4276-7213

***Abstract*—In this paper, we investigate the relationship between multi-agent planning and self-organisation through the combination of two representative approaches both enjoying declarativity. We consider a functional approach to self-organising systems development, called Aggregate Programming (AP), and propose to exploit collective adaptive behaviour to carry out plan revisions.**

***Index Terms*—Multi-agent systems, Robust plan execution, Aggregate computing**

I. Introduction

The task of Multi-Agent Planning (MAP) consists of coordinating the actions of multiple agents in a Multi-Agent System (MAS) towards common goals. Traditionally, planning addresses both the assignment of tasks/actions to individual agents, as well as the coordination among different agents in terms of *causal links* and *concurrent* actions. For large-scale systems of cooperating agents, however, plans can hardly capture, fully and in advance, the behaviour and interactions of each individual agent towards desired global state-of-affairs. Rather, *high-level* plans should be specified and dynamically refined to define both team- and individual-level behaviour in a flexible and adaptable fashion.

Our vision is meant to integrate two approaches to MAS implementation that have followed distinct research paths up to now. On the one hand, there are self-organizing, swarm-based approaches, capable of automatically adapting MAS behaviour based on the contingent situation. Such approaches can be very effective in dealing with small, specific uncertainties in the operating environment, but are not usually suited to drive complex plans made of several phases involving highly heterogeneous behaviours. On the other hand, there are more traditional systems that are able (in principle) to interpret and execute any plan expressed in a suitably standardized language, such as the Multi-Agent-Planning Domain Definition Language (MA-PDDL) [1], but do not have built-in capabilities to exhibit flexible lower-level behaviour and coordination with other agents.

In summary, we aim at investigating the integration of MAP and self-organisation, where the former is key to achieve complex goals requiring possibly long sequences of agents' actions; and the latter is key to deal with the uncertainty of the operating environment via low-level flexibility.

To promote such integration, we further require and focus on *declarative* approaches: the high-level plans should not be hard-coded in the system, but represented in a suitable language; similarly, the self-organizing logic should be expressed through programs abstracting from low-level issues (e.g., inter-agent communication details). We found MA-PDDL and Aggregate Programming (AP) [2], respectively, to be the two main representatives satisfying these requirements.

II. Multi-Agent Planning and Self-Organisation

MAP evolves from one of the oldest AI problems: automated planning of the actions that an agent has to execute in order to reach a *goal* state from starting in an *initial* state. In 2012, the standard language for expressing planning domains and problems (PDDL) has been extended to the MA-PDDL language, which can handle multiple agents [1]. Currently, several planners directly support MAP, both as a centralized and as a decentralized process. As an alternative to using such planners, it is possible to automatically convert a MA-PDDL problem to a single-agent problem that can then be solved with one of the many single-agent planners.

A potential issue with the execution of a MA-PDDL plan is that, if errors occur, it is left unspecified how the agents and the MAS as a whole should react. Some work has been done in the AI community to address monitoring and repair of Multi-Agent Plans (MAPs) (i.e., possibly long and complex sequences of actions that have to be performed by the MAS) [3], however, such approaches only address the monitoring/diagnosis task, assume a centralized monitoring/repair process, or require full/perfect communication between the agents in the team.

The multi-agent community has also followed a somewhat different approach, by identifying a number of problem types that are particularly relevant in practice, such as the *path-planning* problem and the *pickup-and-delivery* problem (i.e., move a set of items from sources to destinations). Such problem types have then been investigated separately, leading to specialized solutions that are often partially hardcoded in the agents behaviours [4].

Self-organisation refers to the process whereby a system autonomously (i.e., without external control) seeks and sustains its order or structures [5]. It is often meant as a bottom-up decentralised process where macro-level structures *emerge* from micro-level activities and interactions. Few programming approaches tailored to self-organising systems exist, with AP being one of the most representative [2]. AP is a paradigm for programming self-organizing systems declaratively by functionally composing global, aggregate behaviours specifying how a set of agents should behave and interact with neighbours. AP is formalised by the Field Calculus (FC) [2], a core language for manipulating *computational fields* [2] which is implemented by full-fledged languages like the Scala-internal *ScaFi (Scala Fields)* [6] and the C++-internal *FCPP* [7].

Our vision is that self-organisation and MAS coordination approaches can be profitably combined. This is related to research efforts, such as *organic computing* [8], promoting ways to balance "creative self-organized bottom-up processes" and "top-down control". Architectural solutions leveraging planning and self-organisation have also been proposed, e.g. in [9] for robotic ensembles. Indeed, self-organisation can be key to promote *continual* planning [10]. However, we adopt an approach based on *declarative programming languages*.

Currently, we focus on supporting resilient execution of MAPs:

- we consider the *plan* (together with declarative models of the actions) as the main force used to drive the (dynamic) structure of a self-organising MAS, by defining workflows and actions to "steer" the self-organisation;
- we advocate that *declarative representations* of MAPs should include both traditional actions (performed by an individual) and collective actions (performed by teams of agents), as required by PDDL extensions;
- we consider the full execution cycle (including monitoring and repair of the plans), which requires both robust and flexible plan execution, and incremental (partial-to-full) re-planning when strictly necessary.

We stress the importance of a declarative representation of the plan (especially the action models) as well as of the monitoring and repair processes (c.f. Section III). We believe that declarativity is essential for decoupling the system specification from execution and deployment issues, as well as for enabling formal analysis including static and dynamic verification.

III. Discussion and Research Roadmap

Ideas about the steps and techniques that may be helpful to achieve our research goals follow.

- A fundamental step would be the development of a language for expressing MAPs. A first step in this direction has been presented in [11], where the authors propose the notion of *aggregate plan* to capture the kind of plans suitable for teams performing collective actions. We still need to formalize a language to express aggregate plans, and the models of the actions involved (in terms, e.g., of pre-conditions, nominal and faulty post-conditions).
- Given an aggregate plan, its execution must be flexible enough. This will require to define a layer exploiting the expressive power of FC to induce such flexible collective behavior in a fully distributed, self-organizing way.
- The properties to be monitored may require more complex mechanisms than just direct observation by individual agents, such as those investigated in existing work on Runtime Monitoring of complex spatial and temporal properties with FC [12]. Furthermore, these properties should be automatically derived from the plan and the actions model, instead of manually specified.
- Finally, we envision a layer for the repair of failures that goes beyond the flexibility directly exhibited by the execution layer. The characteristics of FC and of the systems we address seem to suggest that also such a layer should be an *aggregate process* [13]. In order to avoid hardcoding the repair actions, the layer should exploit knowledge of the plan and of the actions (failure) models in order to update the plan itself. In this way, we will probably start by diagnosing (i.e. assessing) the situation (e.g., several observed delays are due to a congestion) and subsequently finding the minimal plan change required to put the execution back on track.

References

[1] D. L. Kovács, "A multi-agent extension of PDDL3.1," in *International Planning Competition (IPC)*, 2012.

[2] M. Viroli, J. Beal, F. Damiani, G. Audrito, R. Casadei, and D. Pianini, "From distributed coordination to field calculus and aggregate computing," *J. Log. Algebraic Methods Program.*, vol. 109, 2019.

[3] R. Micalizio and P. Torasso, "Cooperative Monitoring to Diagnose Multiagent Plans," *Journal of Artificial Intelligence Research*, vol. 51, pp. 1–70, 2014.

[4] O. Salzman and R. Stern, "Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 1711–1715.

[5] T. De Wolf and T. Holvoet, "Emergence versus self-organisation: Different concepts but promising when combined," in *Engineering Self-Organising Systems, Methodologies and Applications (ESOA workshop, AAMAS conference)*, ser. LNCS, S. Brueckner, G. D. M. Serugendo, A. Karageorgos, and R. Nagpal, Eds., vol. 3464. Springer, 2004.

[6] R. Casadei, M. Viroli, G. Audrito, and F. Damiani, "Fscafi : A core calculus for collective adaptive systems programming," in *ISoLA (2)*, ser. LNCS, vol. 12477. Springer, 2020, pp. 344–360.

[7] G. Audrito, "FCPP: an efficient and extensible field calculus framework," in *International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. IEEE, 2020, pp. 153–159.

[8] J. Branke, M. Mnif, C. Müller-Schloer, H. Prothmann, U. Richter, F. Rochner, and H. Schmeck, "Organic computing - addressing complexity by controlled self-organization," in *Leveraging Applications of Formal Methods (ISoLA)*. IEEE Computer Society, 2006, pp. 185–191.

[9] O. Kosak, C. Wanninger, A. Hoffmann, H. Ponsar, and W. Reif, "Multipotent systems: Combining planning, self-organization, and reconfiguration in modular robot ensembles," *Sensors*, vol. 19, no. 1, p. 17, 2019.

[10] M. Brenner and B. Nebel, "Continual planning and acting in dynamic multiagent environments," *Autonomous Agents and Multi Agent Systems*, vol. 19, no. 3, pp. 297–331, 2009.

[11] M. Viroli, D. Pianini, A. Ricci, and A. Croatti, "Aggregate plans for multiagent systems," *International Journal of Agent-Oriented Software Engineering*, vol. 5, no. 4, pp. 336–365, 2017.

[12] G. Audrito, R. Casadei, F. Damiani, V. Stolz, and M. Viroli, "Adaptive distributed monitors of spatial properties for cyber-physical systems," *Journal of Systems and Software*, vol. 175, 2021.

[13] R. Casadei, M. Viroli, G. Audrito, D. Pianini, and F. Damiani, "Engineering collective intelligence at the edge with aggregate processes," *Engineering Applications of Artificial Intelligence*, vol. 97, p. 104081, 2021.