Combining Self-Organisation with Decision-Making and Planning

Christopher-Eyk Hrabia(✉), Tanja Katharina Kaiser, and Sahin Albayrak

Technische Universität Berlin, DAI-Lab,
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
christopher-eyk.hrabia@dai-labor.de

Abstract. Coordination of mobile multi-robot systems in a self-organised manner is in the first place beneficial for simple robots in common swarm robotics scenarios. Moreover, sophisticated robot systems as for instance in disaster rescue teams, service robotics and robot soccer can also benefit from a decentralised coordination while performing complex tasks. In order to facilitate self-organised sophisticated multi-robot applications a suitable approach is to combine individual decision-making and planning with self-organization. We introduce a framework for the implementation and application of self-organization mechanisms in multi-robot scenarios. Furthermore, the integration into the hybrid behaviour planning framework ROS Hybrid Behaviour Planner is presented. This combined approach allows for a goal-directed application of self-organisation and provides a foundation for an automated selection of suitable mechanisms.

Keywords: Self-organization · Behaviour-based planning
Behaviour networks · Hybrid planning · Decision-making
Multi-robot systems

1 Introduction

An increasing application of mobile, autonomous, and intelligent multi-robot systems in various scenarios can be expected in the near future. Possible application domains range from space exploration, disaster rescue operations with aerial and ground robots to more industrial applications as warehouse logistics or heterogeneous service robots operating as a team in future smart-building environments. To address more complex problems or tasks that need to be distributed amongst several robots, it is necessary to coordinate robots to achieve a collaborative behaviour. An intuitive approach for such collaborative behaviour is the control and coordination of multi-robot systems by a centralised instance. Although, this has disadvantages, like having a single point of failure, a required persistent communication connection and limitations for the number of robots.

This work was partially supported by the German Federal Ministry of Education and Research (BMBF grants 13N14093, project EffFeu).

F. Belardinelli and E. Argente (Eds.): EUMAS 2017/AT 2017, LNAI 10767, pp. 385–399, 2018.
https://doi.org/10.1007/978-3-030-01713-2_27

An alternative might be considering decentralised coordination algorithms that allow to coordinate individual robots, e.g. by applying certain rules on the micro level, which leads to a desired behaviour on the macro level of the system. How to develop and engineer such self-organised systems is commonly researched in the field of self-organisation [5] in general, and in other more specific fields like swarm robotics [7], and swarm engineering [17]. However, most research applying such decentralised approaches in robotics is studying these concepts with very simple robots inspired by social insects without having more sophisticated capabilities or tasks for the individual systems [3].

More sophisticated robots, like service robots or autonomous drones, are using decision-making algorithms or even task-level planning for a more goal-oriented mission control. Decision-making applies methods, like hierarchical state machines [2] or behaviour trees [4]. Task-level planning, for instance applies STRIPS-like planners using the PDDL language [12] or hierarchical task networks (HTN) as used by the well-known SHOP planner [20]. Such approaches are also suitable for centralised multi-robot planning and coordination.

Coordination of mobile multi-robot systems in a self-organised manner is not only beneficial for simple robots in common swarm robotics scenarios. More sophisticated robot systems, for instance in disaster rescue teams, can benefit from decentralised coordination while operating complex tasks. An example would be rescue robots using a self-organised exploration strategy while applying a certain rescue procedure with several dependent tasks to help individual victims once they are found. However, there is a gap in between these worlds of individual or centralised decision-making and planning of robots and self-organised coordination of many simple robots. We are not aware of a system that allows the modelling and implementation of the robot behaviour exhibiting both self-organised coordination as well as task-level decision-making and planning in a common approach. To facilitate the idea of self-organised sophisticated multi-robot systems it is desirable to combine decision-making and planning with self-organisation.

In this paper, we address this problem by first discussing background and related work in Sect. 2. In Sect. 3, we introduce a common framework for the implementation of self-organisation mechanisms in multi-robot scenarios. Furthermore, we show how this framework can be integrated into the hybrid behaviour planning framework – ROS Hybrid Behaviour Planner [16] – in Sect. 4. This combined approach allows for a goal-directed application of self-organisation and provides a foundation for an automated selection of suitable self-organisation mechanisms and configurations. Section 5 introduces a first semi-automated mechanisms selection that can relieve a designer of self-organised multi-robot systems from tedious application-dependent engineering of mechanisms. Finally, we show an experiment as a proof-of-concept in Sect. 6 and summarise our contribution as well as future steps in Sect. 7.

2 Related Work

In the introduction we have already mentioned some popular solutions that have been developed to address well-known challenges in the robotics domain, such as decision making [2,4] and planning [12,20]. In the field of self-organisation much attention is spend on the development of certain mechanisms as surveyed in [24]. Moreover, the development of frameworks that allow for a simplified engineering of multi-robot and self-organised systems is as well in focus. For instance the middleware jSwarm [15] allows for centralized sequential programming with spatial-temporal constraints with concentration on the motion in space of swarm robot systems. jSwarm provides an abstract infrastructure for a centralized controlled distributed swarm robot system. The focus of jSwarm is more on performance optimization with software migration and less on enabling more robust and self-organised distributed systems.

Buzz is a domain specific language that provides a middleware for simplified implementations of swarm robot applications [22]. In Buzz a set of robots (swarm) is a first level object that can be used for group task assignment and set operations (intersection, union, difference, and negation). Furthermore, Buzz has capabilities for neighbourhood operations (queries, filtering, and virtual stigmergy) and information sharing. Nevertheless, the implemented approach does only provide an environment for developers wherein self-organization and cognitive algorithms can be implemented, but neither does it support the selection or evaluation of available algorithms nor the combination with individual decision making and planning.

Fernandez-Marquez et al. [9] analyse, classify and describe a set of bio-inspired self-organizing mechanisms in a domain independent manner. The authors classify mechanisms and their relations into three layers of basic, composed and higher-level patterns. In that sense, the composed layer is created by a combination of basic mechanisms and the higher-level patterns show different options of exploiting the basic and composed mechanisms. On the basic level they identified some basic patterns, such as spreading, aggregation, evaporation, and repulsion forming the foundation for a realization of all composed and higher-level mechanisms. The created catalogue of patterns is intended to be used as a base for modular design and implementation of self-organizing systems. A subset of these described patterns is implemented in the execution model BIO-CORE [10], which provides basic bio-inspired services, namely the basic patterns evaporation, aggregation and spreading as well as the gradient pattern. BIO-CORE consists of three main parts: a shared data space which allows to exchange data, basic bio-inspired services that implement basic bio-inspired mechanisms and interfaces that provide primitives for the agents to interact with the core.

The mentioned frameworks illustrate common research directions in the related fields that focus either on specific self-organisation capabilities, mechanism engineering or providing a common infrastructure for developing self-organised systems. To our best knowledge, there are no approaches that try to combine concepts from decision-making and planning with a general purpose

self-organisation framework as BIO-CORE. In order to provide a foundation for the combination of these two research areas in the domain of multi-robot system we have developed a decision-making and planning framework for the popular Robot Operating System (ROS).[1] The framework ROS Hybrid Behaviour Planner (RHBP) combines the advantages of reactive opportunistic decision-making and goal-oriented deliberative planning in a hybrid architecture [16]. The decision-making layer is based on the idea of behaviour networks that allow for dynamic state transitions and simplify the integration of self-organisation due the heuristic nature of the applied utility-function-based action selection algorithm. The deliberative layer makes use of state-of-the-art planners through its PDDL interface. In RHBP a problem is modelled with behaviours, preconditions, effects and goals, whereby conditions are expressed as a combination of virtual sensors and activation functions. The combined condition objects allow to normalize arbitrary sensor information to create an activation value that is applied in the decision-making process. Here, the decision-making considers the relationship between preconditions and effects as well as the results of the interfaced PDDL-planner. RHBP is the foundation for our work presented in the following sections.

3 Self-organisation Framework

The first contribution of this work is the realisation of a modular and reusable framework for self-organisation. This part of the implementation is completely independent from the RHBP and can be used generally.

A modified version of the bio-inspired design patterns of self-organisation mechanisms presented in [9] constitutes the basis of our self-organisation framework. An advantage of both the original design patterns and our adapted version is their modular character and the modelled relationships between the patterns. Thus, existing patterns can be used as the basis for the realisation of new ones. In contrast to the design patterns presented in [9], our adapted version bases all advanced patterns on the gradient pattern, which allows for a simplified and more general implementation. Furthermore, the patterns were regrouped to categorise them on their purpose instead of their complexity as shown in Fig. 1.

The bio-inspired design patterns are categorised in three groups, namely *Basic Functionality Patterns*, *Movement Patterns* and *Decision Patterns*. Basic Functionality Patterns provide required functionalities for the other pattern categories. Apart from spreading, these patterns do not lead to actions that are executed by the agents themselves. Movement Patterns lead to the movement of the agents, e.g. enabling robots to base their movement on a potential field. Finally, Decision Patterns enable collective decisions.

The central *Basic Functionality Pattern* is the *Gradients* pattern as all *Movement Patterns* and *Decision Patterns* are built on it. Gradients are information which are subject to *Spreading*, *Aggregation* and possibly *Evaporation*. In addition to including all data points required for the advanced patterns, gradients

[1] https://github.com/DAInamite/rhbp.

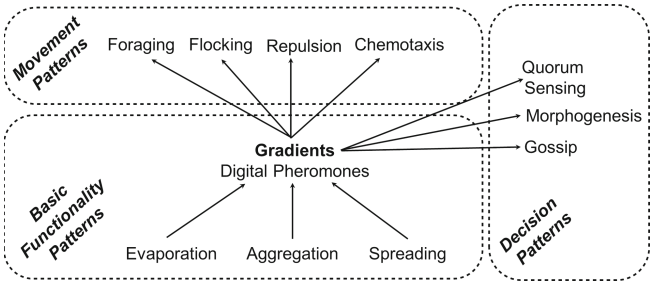


Fig. 1. Bio-inspired self-organisation design patterns. Arrows indicate relationships.

contain a pose indicating where the data is located. Gradients can either be deposited in the environment or be attached to a moving entity like an agent. Moreover, gradients are either spread by agents or present in the environment.

Spread gradient data (*SoMessage*) is stored and manipulated by a library we have named self-organisation buffer (*SoBuffer*). Both Aggregation, or information fusion, and Evaporation, which reduces the relevance of information over time, are applied on the gradient data by the SoBuffer. Each gradient has individual evaporation attributes, namely frequency and strength, attached to it, which specify the rate at which evaporation is executed over the data point by the SoBuffer. In addition, the SoBuffer can aggregate the received gradient data based on their purpose. Hence, gradients which are used for different tasks can be aggregated differently or used without aggregation. For example, gradient data can be aggregated based on its location or its sender. The implementation applies the publish-subscribe design pattern, whereby spread gradient information (*SoMessage*) is received, collected and filtered on the receiver-side (*SoBuffer*). This enables the combination of individual and decentralised mechanisms within one information space. In the following, we consider a self-organisation mechanism as an implementation of an abstract self-organisation pattern.

The *Digital Pheromone* pattern is a special case of the Gradients pattern. Digital Pheromones are evaporating gradients which are deposited in the environment [9]. All Basic Functionality Patterns were realised based on [9].

Advanced patterns can request different subsets of the stored gradient data from the SoBuffer to base their calculations on those. Movement Patterns and Decision Patterns are integrated as mechanisms in the self-organisation framework. Both are based on abstract classes which provide a common basis for the mechanism implementations. Therewith, new mechanisms can be straightforwardly implemented using these blueprints.

Each movement mechanism determines a movement vector an agent can follow. All four Movement Patterns depicted in Fig. 1 are integrated in the self-organisation framework. For each Movement Pattern one or more mechanisms are realised which allow to employ the pattern in different scenarios.

The *Chemotaxis* pattern enables motion coordination based on gradients [9]. Two different algorithms to calculate movement vectors based on gradient fields were implemented to realise this pattern. Firstly, a general approach for attractive and repulsive gradient calculations as in [1] is integrated. Secondly, a more sophisticated gradient calculation was integrated which allows agents to reach an attractive gradient even if it is overlapped by a repulsive gradient following the formulas by [14].

The *Repulsion* pattern enables agents to reach a uniform distribution and to avoid collisions [9]. Two mechanism implementations are provided to realise this pattern. In one mechanism, the repulsive gradient formula of [1] is utilised while the second mechanism applies the repulsion formula presented in [9].

Flocking allows motion coordination and pattern formation in swarms [9]. A mechanism based on [23] is provided and also a more complex version applying the gradient-based formulas of [21] is integrated in the framework.

Moreover, the *Ant Foraging* pattern is part of the presented self-organisation framework. Foraging is a pattern for collaborative search which allows to explore and exploit an environment [9]. The pattern requires several mechanisms to be realised which are based on [9] as well as on [6].

The abstract class for decision mechanisms, which implement the Decision Patterns, includes two common methods. One method determines the current value and state of an agent and depends on the pattern. The other method spreads these values as a gradient message and is universal for all patterns.

Quorum Sensing allows collective decision making based on a required threshold number of agents. It can be implemented in a general way following [9].

The other two Decision Patterns, namely *Morphogenesis* and *Gossip*, are highly dependent on the use case. The Morphogenesis pattern allows to determine the agent's behaviour based on its spatial position [9]. Gossip enables to obtain shared agreements between all agents [9]. For both patterns, sample mechanisms are implemented to exemplify their feasibility. The sample Gossip mechanism determines the maximum spread value while the sample Morphogenesis mechanism determines the barycentre of a group of agents using the algorithm proposed in [18].

The self-organisation framework was realised completely within the Robot Operating System (ROS). Next to being easy to extend and to apply, it provides hardware abstraction and an established messaging communication infrastructure. The latter aspect is useful in particular to realise Spreading. As already mentioned above, the implementation relies on the publisher-subscribe design pattern, which is a core concept of ROS, and it is inspired by the information sharing and filtering architecture of the popular tf package [11].

4 Combining Decision-Making, Planning and Self-organisation

The combination of decision-making, planning and self-organisation is realised by integrating the self-organisation framework presented in Sect. 3 into the RHBP

that is recapped in Sect. 2. The following introduced extension provides necessary components to use the presented self-organisation mechanisms within the hybrid planning structure of the RHBP. The behaviour network layer of the RHBP is based on the dependencies of behaviour preconditions, effects and desired world changes. These dependencies are used for the activation calculation within the RHBP, serving as a heuristic estimation for decision-making and providing a well-matching foundation for the integration of self-organisation. This is because the normalisation of conditions through the application of different utility functions (activation functions) enables the straightforward integration of non-discrete relationships as we find them in self-organisation mechanisms.

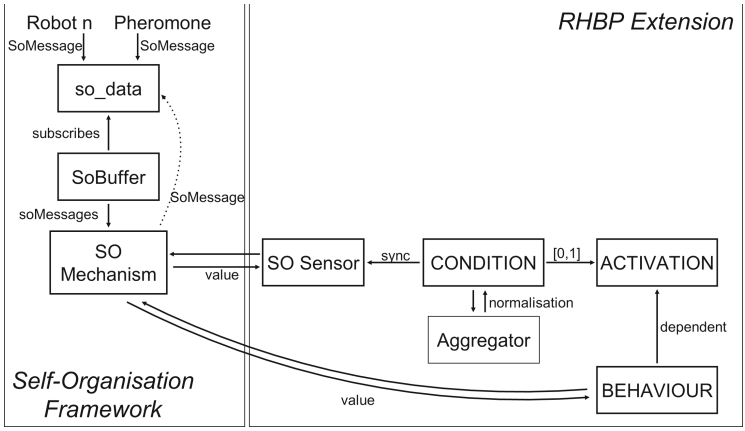


Fig. 2. Architecture for the integration of self-organisation into the RHBP

Figure 2 illustrates both the self-organisation framework presented in Sect. 3 and the integration into the RHBP using components provided by the extension. Three extended component types have been implemented to enable the use of self-organisation mechanisms, namely *sensor*, *condition* and *behaviour*.

The *Self-Organisation Sensor* (*SO Sensor*) is a central component to enable the integration of self-organisation into the RHBP. It is a complex sensor type which senses gradient-based information provided by the self-organisation framework by invoking their common methods. Specifically, it provides access to our SoBuffer and thereby allows to sense movement vectors, gradient values and agent states. Movement vectors are calculated by the movement mechanisms. For some mechanism implementations, it is possible to sense a vector leading to the goal gradient, too. The sensing of values and states of agents are related to the decision mechanisms.

Figure 2 illustrates the usage of the Self-Organisation Sensor in a *Condition*, which normalises the sensor values and maps them to activation levels using the standard activators provided by the RHBP core implementation, for instance

Boolean, threshold and linear functions. Several special self-organisation conditions are provided by the extension to allow the modelling of different application scenarios. For example, conditions related to movement mechanisms determine activation levels based on the presence of a potential field or the length of the movement vector. Sample conditions for mechanisms related to Decision Patterns lead to activation when the state or value of the robot has changed. The provided conditions are not exhaustive and can be extended as required.

The main components for the execution of self-organisation within the RHBP are behaviours. Several behaviours that execute the self-organisation mechanisms provided by the framework are part of the RHBP self-organisation extension. Both movement mechanisms and decision mechanisms are implemented based on abstract classes. Thus, common methods exist for the different mechanisms which can be reused by the specific behaviours to conduct self-organisation.

The *Move Behaviour* executes movement mechanisms within the RHBP by invoking their common method *move()*. The method returns a movement vector which will be transformed into a steering command which matches the robot type. Currently, the extension provides a Move Behaviour which transforms the three dimensional movement vector to a linear velocity in x-direction and an angular velocity around the z-axis. Thus, it is suitable for all differential drive robots. However, providing additional behaviours for other robot types would only require to implement the mentioned conversion from a movement vector to the particular steering command.

The *Decision Behaviour* executes decision mechanisms within the RHBP by invoking their common method *spread()*. This method determines value and state of an agent and spreads those values in a gradient message. The distribution of an agent's value and state is a core aspect for the realisation of Decision Patterns as each agent determines its own value and state based on its neighbours' data.

Several additional behaviours were integrated in the extension to realise behaviours which are not common for all mechanisms. For example, the Ant Foraging Pattern requires that the state of the agents is set to specific values in several cases. Hence, a special RHBP behaviour allows to set the state related to a mechanism to a predefined value.

5 Selecting Self-organisation Mechanisms

Selecting an appropriate self-organisation mechanism for the intended system behaviour is a challenging task. Usually system designers have to choose a suitable coordination mechanism during design time to let multi-robot systems collaborate in the desired fashion to fulfil an intended task. But the suitability of a chosen coordination mechanism might change during task execution as the environment or system capabilities might change. Hence, the *Coordination Mechanism Selector (CMS)* was realised, which provides a foundation to determine the most suitable self-organisation mechanism in a given situation based on expert knowledge or experience and a self-organisation goal that indicates the task of

the agent. Thus, system designers are relieved from the task to select a self-organisation mechanism during design time and it is possible to improve the adaptation capabilities of the resulting system.

The *Self-Organisation Coordinator (SO Coordinator)* consists of two components, namely its own RHBP instance and the Coordination Mechanism Selector, as illustrated in Fig. 3. Self-organisation mechanisms can be encapsulated by the Self-Organisation Coordinator as all components required for the realisation of a self-organisation mechanism, e.g. behaviours and goals, are assigned to its own RHBP instance. Thus, a higher-level RHBP instance can treat the self-organisation mechanism as one behaviour, in the form of the Self-Organisation Coordinator, no matter how many components are required for its realisation. Hence, its own RHBP instance is used to monitor and control the particular self-organisation mechanism realised within our framework. It is also possible to integrate or combine several self-organisation mechanisms by using multiple Self-Organisation Coordinator instances per Behaviour Network. This architecture helps to separate the application specific modelling using the RHBP from a generic self-organisation mechanism implementation and makes the mechanism implementation exchangeable.

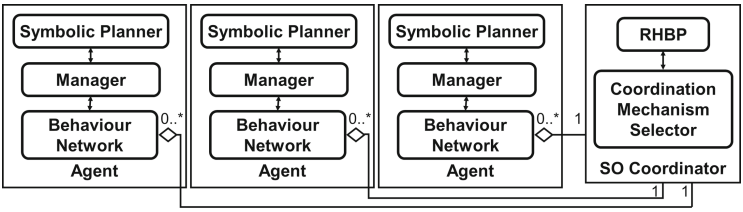


Fig. 3. Integration of the Coordination Mechanism Selector in the structure of the RHBP. A SO Coordinator is always bound to an individual agent in a decentralised fashion.

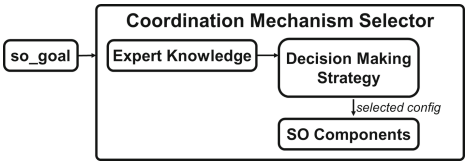


Fig. 4. Architecture of the Coordination Mechanism Selector

As illustrated in Fig. 4, the Coordination Mechanism Selector consists of three layers, namely *Expert Knowledge*, *Decision Making Strategy* and *Self-Organisation Components (SO Components)*. Additionally, each Self-Organisation Coordinator requires a self-organisation goal, which indicates the task of the multi-robot system, as input.

The Expert Knowledge maps self-organisation goals to a list of suitable mechanism options. Each option consists of a *configuration key*, a *score* and *parameters*. The configuration key is an indicator for a self-organisation configuration

that can be used to fulfil a self-organisation goal. The indicated self-organisation configurations are specified in a reusable configuration library. Each configuration includes RHBP components and parameters as presented in Sects. 3 and 4. The parameters of the configuration will be replaced with the parameters included in the option to adjust the setting based on the self-organisation goal. The score specified in each option rates the feasibility of the self-organisation goal using the specified self-organisation configuration. This allows to create a repository of suitable self-organisation configurations that have been evaluated in respect to a given system goal. This concept is inspired by the proposed hypothesis database of Edmonds et al. [8].

The Decision Making Strategy is the central component of the Coordination Mechanism Selector. Its aim is to determine a suitable self-organisation configuration based on a self-organisation goal and the provided Expert Knowledge. Moreover, it might be used to adjust the score included in the options of the Expert Knowledge, e.g., using online learning [19] or evolutionary approaches [13]. Therewith, the decision making process can incorporate experience, which is essential to determine the most suitable self-organisation mechanism in a dynamic environment. As the environmental conditions might change during task execution, the Decision Making Strategy will re-evaluate its decision and adjust the self-organisation mechanism during task execution if required. In the current development stage, the Decision Making Strategy selects the option with maximum score, which is an empirically determined value for a given self-organisation goal.

After determining a configuration key for a self-organisation configuration and its parameter adjustments, the Self-Organisation Components factory will create the specified RHBP components. These components are associated with the RHBP instance being part of the Self-Organisation Coordinator and are therewith encapsulated. All three layers of the Coordination Mechanism Selector can be replaced or enhanced straightforwardly due to its modular structure.

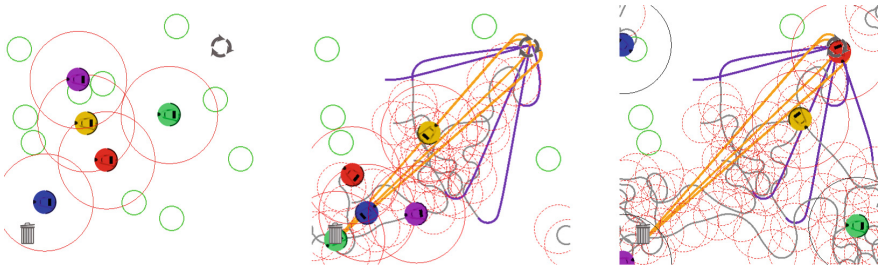
6 Experiment

In this section, we illustrate the application of our solution with a simulated example scenario. The presented experiment serves as a first proof of our work. The example scenario is comprised of multiple robots that have to maintain an open unknown space by keeping it clean and managing the recycling and dumping process of found garbage items. The recycling and dumping process requires that once garbage is found, first it needs to be transported to a recycling station, before all leftovers are transported to the dump station. The robots have to patrol the environment repeatedly over time. Moreover, the robots have to make sure that they avoid collisions with each other.

The cleaning process with several dependent stages is different to common swarm robot experiments that focus on achieving one certain stable state in a decentralised manner. We have integrated this additional complexity in order to illustrate the beneficial combination of self-organisation with more complex

decision-making and planning. However, we still keep this simulated experiment comparably simple to improve the transparency.

For the simulation, we use an extended version of the basic turtlesim simulation, commonly known from the ROS beginner tutorials. This simulation allows to control multiple differential wheeled robots in an empty space. Our version[2] extends the original implementation with capabilities of sensing other robots, allowing to configure a torus environment without borders, adding various visualization options to draw additional elements into the world and replacing the turtle robots with cleaning robots. To simulate the garbage items and their detection we make also use of our SoBuffer implementation to randomly spread garbage gradients with a special identifier in the environment, which then can be sensed with the corresponding gradient sensors. The simulation environment in a particular start configuration and during execution is shown in Fig. 5.



(a) Start (b) Execution 1 (c) Execution 2

Fig. 5. Visualised simulation scenario. Green circles are garbage items; Red solid circles around robots denote the sensor range. Red dotted cycles show virtual pheromones, the size corresponds to the evaporation stage (smaller=older). Garbage bin and recycling symbol visualize dump and recycling station. Grey lines mark robot trajectories; Purple lines mark trajectories after garbage was collected; Orange lines mark trajectories after garbage was recycled. (Color figure online)

The modelled RHBP solution with behaviours, goals and corresponding preconditions and effects is visualised in Fig. 6. Each robot instantiates this model independently resulting in a decentralised solution with coordination and interaction amongst the robots only carried out through the simulation environment by sensing each other as well as exchanging information through the virtual gradient space of the SoBuffers. Three goals formulate the target conditions for the robots, *PatrolEnvironment* expressing the need for a repeated cleaning process in the environment, *GarbageCleaned* modelling the need to clean garbage items once they are found, and *AvoidCollision* to keep the robots in a safe distance of each other. The garbage recycling and dumping state is tracked by influencing special ROS topics in the related behaviours and accessing them through so

[2] https://github.com/cehberlin/ros_tutorials/tree/clean_robots.

called *KnowledgeSensors*. The picking of garbage is implemented by sending gradient information that result in the deletion of garbage gradients at this position. The behaviours *MoveToDump* and *MoveToRecycling* are using distance conditions formulated with *LinearActivators*, which provide higher activation for a larger distance to target, and *Pose* sensors for the current robot position.

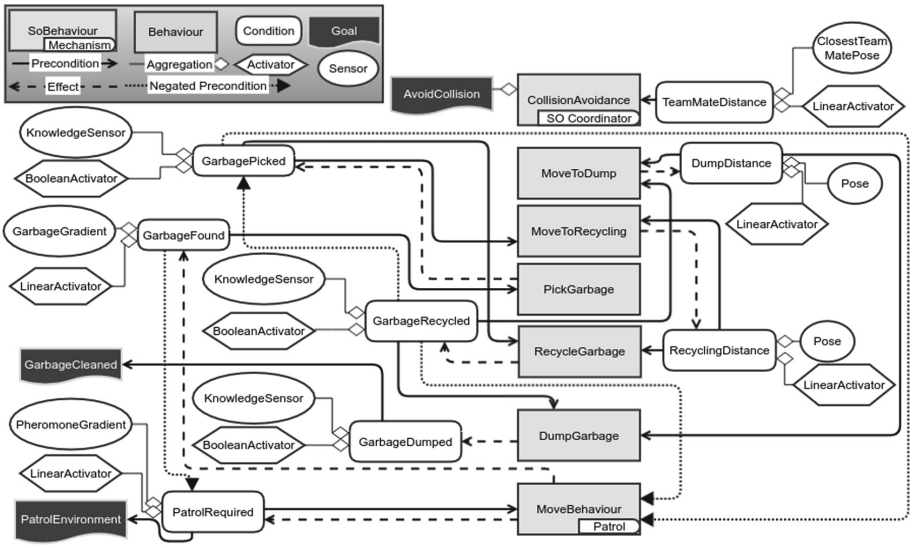


Fig. 6. Model of the RHBP solution for an individual robot of the experiment illustrating the relationships with preconditions, behaviours and effects.

So far this part of the model is expressed with RHBP core components. Nevertheless, the exploration and patrolling applies self-organisation with a patrol mechanism that is based on the virtual pheromone pattern. In our implementation each robot spreads evaporating gradients at its current position while moving through the environment and calculates the movement vector using a repulsion pattern to push itself away from the gradient field. Both together results in robot motion that prefers the motion into unknown space or space that has not been visited for a longer time period. All robots share these virtual pheromones through the SoBuffer library communication infrastructure, thus they are able to coordinate in a self-organised manner.

In the shown model, we have two self-organisation mechanisms for combined exploration-patrol and collision avoidance. For collision avoidance we apply the *So Coordinator* that automatically selects a suitable mechanism based on the given self-organisation goal *AvoidCollision* and the available scores in our database. The different characteristic of the goal is also indicated by the visualised aggregation relationship instead of a link to a condition. In contrast to collision avoidance, patrolling is manually selected by integrating directly the

Patrol mechanism. The direct selection of a mechanism has the disadvantage of making a later exchange of the mechanism more difficult. However, we have taken this approach here to illustrate different possible usage styles of our framework, although the specific *Patrol* mechanisms could be replaced easily by an *SO Coordinator* instance with a corresponding self-organisation goal.

The scenario visualised in Fig. 5 with the model of Fig. 6 has been tested with 5 robots and 10 garbage items randomly positioned in 5 different start configurations (scenarios). The positions of recycling and dump station have not been altered between the trials. Moreover, we have manually manipulated the expert knowledge scores to force the *SO Coordinator* to run all 5 scenarios once with the collision avoidance mechanisms based on the repulsion pattern from Fernandez-Marquez et al. [9] and once with the algorithm from Balch and Hybinette [1]. Both patterns rely on robot pose gradients to allow the robots to determine the repulsive forces from each other.

Table 1. Experiment results of a comparison between different available self-organisation patterns for collision avoidance

	Fernandez-Marquez		Balch/Hybinette	
Scenario	Duration in s	Collisions/s	Duration in s	Collisions/s
1	270.39	2.02	89.84	2.65
2	1484.70	1.42	177.26	1.42
3	1162.18	1.60	471.90	1.55
4	952.40	1.22	326.14	2.08
5	363.93	2.25	267.02	3.11
Mean	846.72	1.70	266.43	2.16
Median	952.40	1.60	267.02	2.08
STD	465.39	0.38	130.32	0.64

The experiment results are listed in Table 1 and show different characteristics for both applied mechanisms. We see that the runs with Balch and Hybinette mechanism clearly outperform runs with the Fernandez-Marquez mechanism in terms of execution time (duration) for completing the mission. However, runs with Fernandez-Marquez mechanism are having fewer collisions per time. For the number of collisions over time, it has to be considered that one collision might be counted several times, depending on the sensor frequency of the simulation, if the robots stay for a moment in the collision pose.

The obtained results of this experiment could now be used to create a score for the mechanism configurations in our expert knowledge library. However, we have not yet fine-tuned the parameters of the mechanisms, which might influence the results. Thus it would be useful to repeat the experiment with other parameter configurations. Nevertheless, the presented results illustrate the application

of our work and highlight the importance of an inexpensive exchange of self-organisation mechanisms and their configuration within a decision-making and planning framework. However, it also indicates that our SO coordinator concept is providing a suitable foundation for the integration of experience and expert knowledge with self-organisation pattern, but a system designer needs further support on an automatic determination of the mechanisms scores.

7 Conclusion

In this paper, we have presented a general purpose self-organisation library for the multi-robot domain and how this framework can be combined with our decision-making and planning framework RHBP. The self-organisation library (SoBuffer) already provides a wide range of mechanisms and allows for additional extensions due to its modular and generic architecture. The further introduced integration of the library into the RHBP closes a gap between the research fields of decision-making, planning and self-organisation. It enables the development of autonomous robots resolving tasks with complex dependencies while allowing for self-organised coordination and problem resolving within one framework. Both worlds can directly interact with each other by making use of the same domain model, information sources and abstract system capabilities. The integration into the popular ROS framework does also guarantee a fast adoption and integration into existing solutions and software ecosystems.

Furthermore, we have extended our self-organisation library with the concept and an initial implementation of SO Coordinator components that help to abstract the actual self-organisation mechanisms from the intentions of a system designer. Moreover, this approach provides the necessary infrastructure to collect abstract self-organisation mechanisms and their configuration in a common database. While the current determination of the mechanism scores is only making use of expert experience, it already includes infrastructure to enable more sophisticated approaches in the near future, like online and offline machine learning or applying evolutionary strategies. Besides proceeding towards this direction, we plan to further evaluate our solution in real life experiments using multi-robot systems.

References

1. Balch, T., Hybinette, M.: Social potentials for scalable multi-robot formations. In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2000, vol. 1, pp. 73–80. IEEE (2000)
2. Bohren, J., Cousins, S.: The SMACH high-level executive [ROS news]. IEEE Robot. Autom. Mag. **17**(4), 18–20 (2010)
3. Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M.: Swarm robotics: a review from the swarm engineering perspective. Swarm Intell. **7**(1), 1–41 (2013)
4. Colledanchise, M., Ögren, P.: How behavior trees modularize hybrid control systems and generalize sequential behavior compositions, the subsumption architecture, and decision trees. IEEE Trans. Robot. **33**(2), 372–389 (2017)

5. De Wolf, T., Holvoet, T.: Emergence versus self-organisation: different concepts but promising when combined. In: Brueckner, S.A., Di Marzo Serugendo, G., Karageorgos, A., Nagpal, R. (eds.) ESOA 2004. LNCS (LNAI), vol. 3464, pp. 1–15. Springer, Heidelberg (2005). https://doi.org/10.1007/11494676_1
6. Deneubourg, J.-L., Aron, S., Goss, S., Pasteels, J.M.: The self-organizing exploratory pattern of the argentine ant. J. Insect Behav. **3**(2), 159–168 (1990)
7. Dorigo, M., et al.: Evolving self-organizing behaviors for a swarm-bot. Auton. Robot. **17**(2–3), 223–245 (2004)
8. Edmonds, B.: Using the experimental method to produce reliable self-organised systems. In: Brueckner, S.A., Di Marzo Serugendo, G., Karageorgos, A., Nagpal, R. (eds.) ESOA 2004. LNCS (LNAI), vol. 3464, pp. 84–99. Springer, Heidelberg (2005). https://doi.org/10.1007/11494676_6
9. Fernandez-Marquez, J.L., Di Marzo Serugendo, G., Montagna, S., Viroli, M., Arcos, J.L., Arcos, J.L.: Description and composition of bio-inspired design patterns: a complete overview. Natural Comput. **12**(1), 43–67 (2013)
10. Fernandez-Marquez, J.L., Serugendo, G.D.M., Montagna, S.: BIO-CORE: bio-inspired self-organising mechanisms core. In: Hart, E., Timmis, J., Mitchell, P., Nakamo, T., Dabiri, F. (eds.) BIONETICS 2011. LNICST, vol. 103, pp. 59–72. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32711-7_5
11. Foote, T.: tf: the transform library. In: 2013 IEEE International Conference on Technologies for Practical Robot Applications (TePRA), Open-Source Software Workshop, pp. 1–6, April 2013
12. Fox, M., Long, D.: PDDL2.1: an Extension to PDDL for expressing temporal planning domains. J. Artif. Int. Res. **20**(1), 61–124 (2003)
13. Francesca, G., et al.: AutoMoDe-Chocolate: automatic design of control software for robot swarms. Swarm Intell. **9**(2–3), 125–152 (2015)
14. Ge, S.S., Cui, Y.J.: New potential functions for mobile robot path planning. In: Proceedings of the 14th IFAC World Congres, pp. 509–514 (1999)
15. Graff, D., Richling, J., Werner, M.: jSwarm: distributed coordination in robot swarms. In: Robotic Sensor Networks (RSN) (2014)
16. Hrabia, C.-E., Wypler, S., Albayrak, S.: Towards goal-driven behaviour control of multi-robot systems. In: 2017 3nd International Conference on Control, Automation and Robotics (ICCAR), pp. 166–173, April 2017
17. Kazadi, S.T.: Swarm engineering. phd, California Institute of Technology (2000)
18. Mamei, M., Vasirani, M., Zambonelli, F.: Experiments of morphogenesis in swarms of simple mobile robots. Appl. Artif. Intell. **18**, 903–919 (2004)
19. Matarić, M.J.: Issues and approaches in the design of collective autonomous agents. Robot. Auton. Syst. **16**(2–4), 321–331 (1995)
20. Nau, D.S., et al.: Shop2: an HTN planning system. J. Artif. Intell. Res. **20**, 379–404 (2003)
21. Olfati-Saber, R.: Flocking for multi-agent dynamic systems: algorithms and theory. IEEE Trans. Autom. Control. **51**(3), 401–420 (2006)
22. Pinciroli, C., Beltrame, G.: Buzz: an extensible programming language for heterogeneous swarm robotics. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3794–3800. IEEE (2016)
23. Reynolds, C.W.: Steering behaviors for autonomous characters. In: Game Developers Conference 1999, pp. 763–782 (1999)
24. Serugendo, G.D.M., Gleizes, M.P., Karageorgos, A.: Self-organisation and emergence in MAS: an overview. Informatica (Slovenia) **30**(1), 45–54 (2006)