Swarm Intell (2015) 9:1–22
DOI 10.1007/s11721-014-0102-6

The TAM: abstracting complex tasks in swarm robotics research

Arne Brutschy · Lorenzo Garattoni · Manuele Brambilla · Gianpiero Francesca · Giovanni Pini · Marco Dorigo · Mauro Birattari

Received: 9 March 2014 / Accepted: 17 December 2014 / Published online: 25 January 2015

Abstract Research in swarm robotics focuses mostly on how robots interact and cooperate to perform tasks, rather than on the details of task execution. As a consequence, researchers often consider abstract tasks in their experimental work. For example, foraging is often studied without physically handling objects: the retrieval of an object from a source to a destination is abstracted into a trip between the two locations—no object is physically transported. Despite being commonly used, so far task abstraction has only been implemented in an ad hoc fashion. In this paper, we propose a new approach to abstracting complex tasks in swarm robotics research. This approach is based on a physical device called the "task abstraction module" (TAM) that abstracts single-robot tasks to be performed by an e-puck robot. A complex multi-robot task can be abstracted using a group of TAMs by first modeling the task as the set of its constituent single-robot subtasks and then abstracting each subtask with a TAM. We present a collection of tools for modeling complex tasks, and a framework for controlling a group of TAMs such that the behavior of the group implements the model of the task.

Guest editor: Roderich Groß.

A. Brutschy (✉) · L. Garattoni · M. Brambilla · G. Francesca · G. Pini · M. Dorigo · M. Birattari
IRIDIA, Université Libre de Bruxelles, Brussels, Belgium
e-mail: arne.brutschy@ulb.ac.be

L. Garattoni
e-mail: lgaratto@ulb.ac.be

M. Brambilla
e-mail: mbrambil@ulb.ac.be

G. Francesca
e-mail: gianpiero.francesca@ulb.ac.be

G. Pini
e-mail: gpini@ulb.ac.be

M. Dorigo
e-mail: mdorigo@ulb.ac.be

M. Birattari
e-mail: mbiro@ulb.ac.be

The TAM enables research on cooperative behaviors and complex tasks with simple, cost-effective robots such as the e-puck—research that would be difficult and costly to conduct using specialized robots or ad hoc task abstraction. We demonstrate how to abstract a complex task with multiple TAMs in an example scenario involving a swarm of e-puck robots.

Keywords Swarm robotics · Task abstraction · Swarm intelligence · Robotics

1 Introduction

Research in swarm robotics focuses on how robots interact and cooperate to perform tasks (Beni 2005; Dorigo et al. 2014), rather than on the details of task execution. For example, consider a hypothetical swarm-operated assembly line where one group of robots drills holes through several parts and another group subsequently bolts them together; or imagine a swarm of nano-bots that extirpate cancer cells: one group of robots identifies and marks tumors; another group subsequently destroys them. While task execution is completely different in the two examples, the logical relationship between the tasks is the same: two tasks have to be executed one after the other. If the focus of the research is to develop coordination mechanisms that allow a swarm to tackle tasks with this kind of logical relationship, it might be desirable to isolate the logical relationship from the details of task execution and focus on the logical relationship, rather than spending resources on inessential aspects of the implementation. We call task abstraction the process by which one focuses on the logical relationship between tasks and omits the details of their execution.

Task abstraction is not a novel concept in swarm robotics research; in fact, it has been used implicitly in numerous studies—for a comprehensive review of the swarm robotics literature, see Brambilla et al. (2013). However, up to now, task abstraction was either (i) confined to simulation or (ii) conducted using some sort of ad hoc solution. Simulation has the advantage of being inexpensive, but approaches developed solely in simulation may suffer from the so called "reality gap" (Jakobi et al. 1995; Francesca et al. 2014b). This is particularly relevant in complex systems, where small but unavoidable differences between simulation and reality could lead to widely diverging behaviors.

Ad hoc solutions are task abstractions that are created for use in the context of a specific experiment. Typically, ad hoc solutions are not generalizable as they are tightly connected to the nature of the experiment. As a result, they cannot be easily and directly exploited in other experiments. Examples of ad hoc solutions to task abstractions are: using travel between two locations to abstract the act of physically transporting objects (e.g. Kernbach et al. 2012; Acerbi et al. 2007; Francesca et al. 2014a, b); using tailor-made inanimate physical objects to abstract manipulation tasks (e.g., Ijspeert et al. 2001); and using special electronic devices to abstract tasks with some dynamic property (e.g. Matarić et al. 2003). Due to their impromptu nature, ad hoc solutions are typically only used to abstract simple tasks that can be tackled by a single robot without any relations to other robots or tasks. Indeed, tasks that require multiple robots are much harder to abstract due to the interrelationships between the constituent single-robot subtasks and the actions of the robots. Additionally, experiments that use ad hoc solutions are costly and difficult to replicate by other researchers—a fact that effectively limits the complexity of the tasks studied in the literature.

In this paper, we propose a new approach to abstracting complex multi-robot tasks in swarm robotics research. This approach is based on a physical device that serves as an abstraction of single-robot tasks to be performed by an e-puck robot. We call this device the TAM, an acronym for *task abstraction module* (see Fig. 1). In abstract terms, we say that a robot performs a single-robot task if it is busy for a given amount of time at a specific location

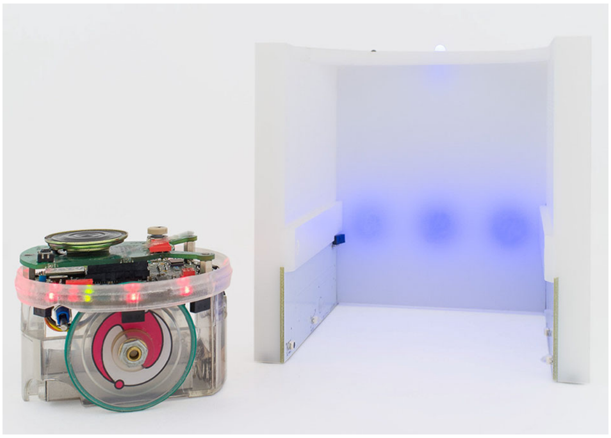


Fig. 1 The TAM and the e-puck robot. The TAM is a booth into which an e-puck can enter. Once an e-puck has entered fully into the TAM, it is considered to be working on the task abstracted by the TAM. The behavior of multiple TAMs can be coordinated to implement interrelationships between the tasks they abstract

and at a specific moment in time. A TAM implements this abstraction of single-robot tasks for real-robot experiments. The TAM operates at an intermediate level of abstraction between simulation experiments and robot experiments. As such, the TAM can be considered a task emulator similar to hardware emulators used in integrated circuit design.

Complex multi-robot tasks can be abstracted by a group of TAMs as follows. First, a complex task is modeled as the set of its constituent single-robot subtasks and their interrelationships. Second, each single-robot subtask is abstracted by a single TAM and the behavior of the TAMs is coordinated such that it reflects the interrelationships identified by the model. We present a collection of tools for modeling complex tasks and a framework for controlling a group of TAMs such that the behavior of the group implements the model of the complex task. The combination of the TAM, the modeling tools, and the control framework forms a new approach for conducting research in swarm robotics, which enables research on cooperative behaviors and complex tasks with simple, cost-effective robots such as the e-puck.[1] The research enabled by the TAM would be difficult and costly to conduct using specialized robots or ad hoc task abstraction.

This paper is organized as follows. In Sect. 2, we describe how to abstract a single-robot task using the TAM. In Sect. 3, we describe how to abstract a complex multi-robot task using a group of TAMs. In Sect. 4, we demonstrate the use of the TAM in a real-robot scenario. In Sect. 5, we review the literature according to the task studied. In Sect. 6, we summarize the contributions of this work and present some directions for future research.

2 Using a TAM to abstract a stationary atomic task

In this paper, we call a task that can be performed by a single robot in a defined time window an *atomic task*. We call a task that has to be performed at a specific location a *stationary task*. Finally, we call a task that is both atomic and stationary a *stationary atomic task*. In other words, we say that a robot performs a stationary atomic task if it is busy for a given

[1] The e-puck is a small, round mobile robot designed for research purposes by Mondada et al. (2009).

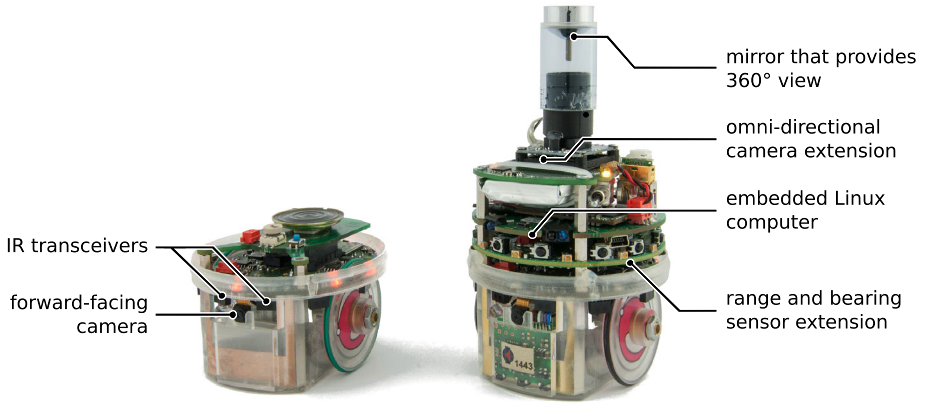

mirror that provides 360° view
omni-directional camera extension
embedded Linux computer
range and bearing sensor extension
IR transceivers
forward-facing camera

Fig. 2 Two configurations of the e-puck robot. The robot on the left is an e-puck without any extension. The sensors relevant to the TAM are the forward-facing camera for the detection of the TAM and the IR transceiver for communication with the TAM. The robot on the right is an e-puck with several extensions: the range and bearing sensor (Gutiérrez et al. 2008), the embedded computer running Linux (http://www.gctronic.com/) and the omni-directional camera. The TAM is compatible with all these extensions; the omni-directional camera is used to detect the TAM instead of the forward-facing camera, if present. We use robots configured as shown on the right in the demonstration presented in Sect. 4

amount of time at a specific location and at a specific moment in time. Examples of stationary atomic tasks are “push a button”, “hold a door open for a given amount of time”, and “guard a specific location”.

2.1 TAM concept and design goals

We conceived the TAM as a physical device that abstracts stationary atomic tasks for laboratory experiments involving the e-puck robot. The e-puck is a mobile robot designed for educational and research purposes.[2] It is small, compact, extensible, and relatively cheap (Mondada et al. 2009). The e-puck is well suited for swarm robotics research as attested by the large number of studies that have been conducted using it (for example, Sperati et al. 2008; Campo et al. 2011; Francesca et al. 2014b; Gauci et al. 2014). Figure 2 illustrates the e-puck and describes the sensors relevant to the TAM.

Physically, the TAM is a booth into which an e-puck can enter. The TAM is equipped with RGB LEDs, light barriers, and an IR transceiver for communication. The TAM can use its RGB LEDs to announce the availability of work, that is, the presence of the stationary atomic task it abstracts. Different tasks can be signaled by using different LED colors.

The light barriers allow the TAM to detect when a robot enters into it. The IR transceiver can be used to communicate with a robot inside the TAM. See Fig. 3a for a conceptual drawing of the TAM.

An e-puck can perceive the RGB LEDs of the TAM using its color camera. If an e-puck perceives a TAM in its proximity, it can decide to work on the stationary atomic task abstracted by that TAM by entering into it. Upon detection of the robot, the TAM reacts according to a user-defined logic; for example, by changing the color of its LEDs, communicating with the robot, or sending information to other TAMs.

[2] http://www.e-puck.org/.

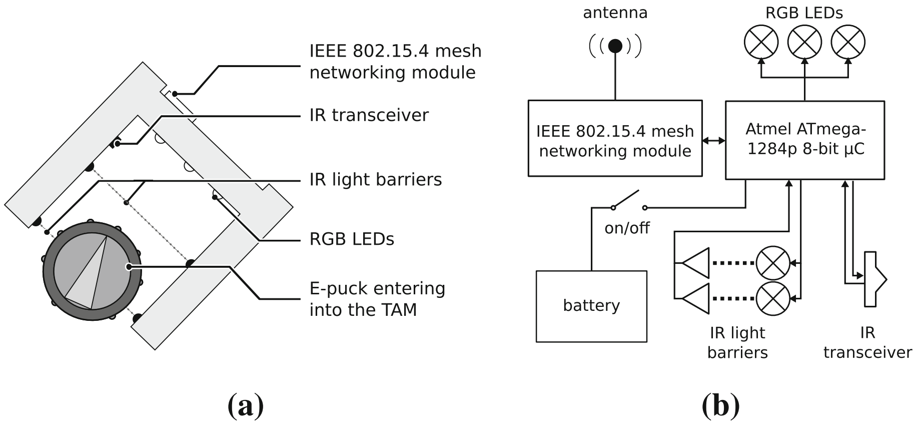

IEEE 802.15.4 mesh networking module
IR transceiver
IR light barriers
RGB LEDs
E-puck entering into the TAM
(a)
antenna
RGB LEDs
IEEE 802.15.4 mesh networking module
Atmel ATmega-1284p 8-bit µC
on/off
battery
IR light barriers
IR transceiver
(b)

Fig. 3 **a** Schematic drawing of the TAM. **b** Block diagram showing the functional components of the TAM

Our goal is to design the TAM so that it is remotely controllable by a central computer, it can report experimental data to the computer, it operates without being physically tethered to the computer, and it is considerably cheaper than an e-puck robot.

2.2 Implementation of the TAM

The TAM is based on *Arduino*,[3] an open-source embedded electronics platform that uses an Atmel microcontroller of the AVR family as a central processor. Arduino is widely available, supported by a large community, and easier to use than other embedded electronics platforms (Banzi 2008). Figure 3b shows a block diagram of the functional components of the TAM.

The TAM adopts an 8-bit RISC processor (ATmega-1284p, 16 MHz), which runs the firmware that locally controls the behavior of the TAM. The RGB LEDs support 24-bit colors and are diffused by a sheet of semi-transparent plastic to facilitate detection by the e-puck. An e-puck can perceive the LEDs of the TAM only from an acute angle: experiments have shown that e-pucks can detect the TAM when positioned in a cone-shaped area that extends up to a distance of 90 cm in front of the TAM's opening (see Fig. 4).

As mentioned above, the TAM announces different task types using different LED colors. Hence, the number of different tasks that the TAM can announce depends on the capability of the robots to differentiate between LED colors. In case of an e-puck equipped with the omni-directional camera extension, we were able to differentiate between up to six tasks.

Communication between the TAM and an e-puck is implemented using the IR transceiver and the e-puck library *IRcom*.[4] This functionality can be used to further differentiate tasks once the robot is inside the TAM.

Autonomy is facilitated by a rechargeable lithium-ion battery with 5 Wh capacity. A single battery lasts over 10 h in a typical experiment (we assume the demonstration presented in Sect. 4 to be a typical experiment).

The TAM is equipped with a IEEE 802.15.4 wireless mesh network module. The TAM can be configured to work on 4 different wireless channels, which allows up to four experiments

[3] http://www.arduino.cc/.

[4] http://gna.org/projects/e-puck/.

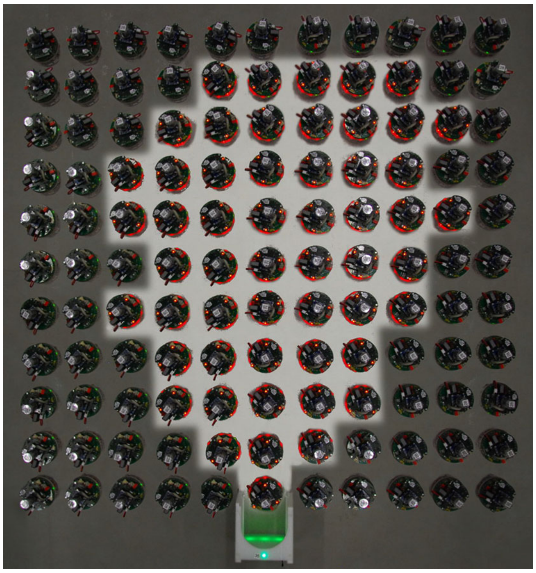


Fig. 4 Area in which an e-puck is able to perceive the TAM when using the omni-directional camera extension. To obtain this image, an e-puck was placed with random orientations on a 10 cm by 10 cm grid. The e-puck signaled that it can perceive the TAM at a given location by setting its LEDs to *red*. The asymmetry in the perception area results most likely from imperfections in the mirror of the omni-directional camera (Color figure online)

to be run in parallel. The TAMs involved in each experiment would operate on the same channel and would not interfere with the TAMs of the other experiments.

The TAM has a cubical shape with a length of 12 cm in every dimension. We designed the body of the TAM so that an e-puck can enter into the TAM without accidentally moving it. This is achieved by two measures: first, we chose a material for the body that is relatively heavy (POM or polyoxymethylene plastic), and second, we equipped the TAM with six rubber feet that increase friction between the TAM and the floor. The TAM can be used with the standard e-puck (without extensions) as well as with an e-puck using the Linux extension and the omni-directional camera.

The TAM is fully supported by the ARGoS simulation framework (Pinciroli et al. 2012), which simulates the whole set of sensors and actuators available on the TAM and the e-puck. For both devices, the ARGoS framework enables the usage of software developed in simulation on the real device without requiring any changes.

The TAM is open source under the *Creative Commons Attribution-Share-Alike 3.0 Unported License*. The TAM possesses an extension connector that allows researchers to extend its capabilities. For further details regarding this connector, the implementation of the TAM, including all data required for production (circuit schematics, layouts of the circuit boards, CAD-models of the body, and the firmware) as well as the control framework required for controlling a large number of TAMs (cf. Sect. 3), see the supplementary online material of this paper (Brutschy et al. 2013) and the accompanying technical report (Brutschy 2014).

2.3 Control framework

We propose a centralized framework for controlling groups of TAMs. In an experiment, the control framework governs the state of each TAM and changes it according to interrelationships between the stationary atomic tasks that the TAMs abstract. Note that this control framework does not control the robots of the swarm, which remains a fully distributed system.

The control framework consists of two components: the *coordinator* and the *firmware* running on each TAM—see Fig. 5 for a graphical representation. The coordinator controls each individual TAM by issuing commands to and receiving event notifications from the TAM's

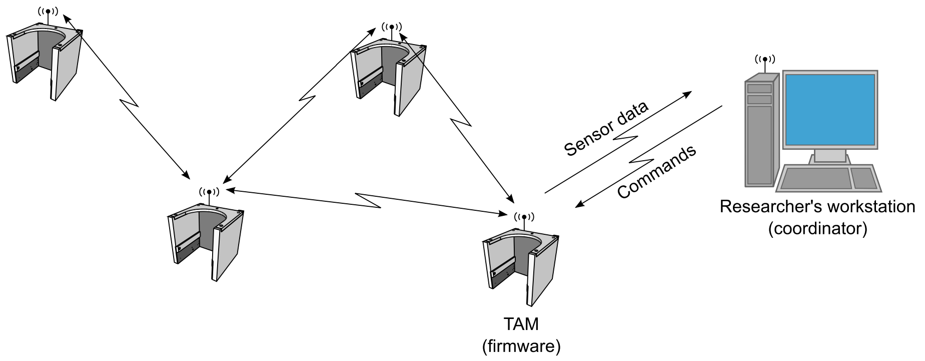


Fig. 5 The control framework and its components. The central coordinator remotely controls the TAMs in an experiment by issuing commands to and receiving event notifications from the firmware running on each TAM. Commands and event notifications are relayed by the wireless mesh network modules of the TAMs

firmware. The firmware reports all events and changes in sensory data to the coordinator, and executes all commands that it receives in return. Commands and event notifications are relayed using the wireless mesh network modules of the TAMs.

The control framework that we propose has several advantages. First, it makes setting up and conducting experiments with TAMs relatively effortless, as changing the behavior of all TAMs requires only to modify the coordinator. Second, the central design allows for accurate statistics-keeping during experiments: all events can be recorded using a central time, which is required for the consistency of the experimental records. This, in turn, allows researchers to fuse data from multiple TAMs with external sensor data (e.g., from a tracking system).

2.4 Reliability experiments

We conduct two experiments in order to evaluate the reliability of the TAM device, the mesh network, and the control framework. Both experiments have been recorded using a video camera. Additionally, we recorded all the available data from the TAMs using the coordinator. Videos and data are available in the supplementary online material (Brutschy et al. 2013).

The goal of the first experiment is to measure the reliability and battery life of the TAM. In the experiment, a single e-puck robot has to continuously perform two stationary atomic tasks abstracted using two TAMs. The TAMs are positioned in the arena such that they are facing each other with a distance of 50 cm. The robot has to alternate between tasks; each task is abstracted using a single TAM. The experiment is terminated once the battery of the robot is depleted. We conducted a single trial that terminated after 40 min. During this time, the robot executed a total of 96 single-robot tasks. No failures occurred during the runtime of the experiment.

The goal of the second experiment is to evaluate the mesh network and the control framework in terms of scalability. In the experiment, all 50 TAMs available to us are used to abstract 50 atomic tasks. The TAMs are positioned in the arena following a decagon shape with 5 TAMs on each side and a diameter of 184 cm. The tasks have to be performed by two e-puck robots. The experiment terminates once each of 50 atomic tasks has been performed exactly once. We conducted a single trial that terminated after 17 min. Again, no failures occurred during the runtime of the experiment.

In addition to these experiments, we can consider the data recorded during the demonstrations presented in Sect. 4. The data shows that during these demonstrations, the robots performed 35 complex tasks successfully. Two tasks failed because robots abandoned a task due to sensor noise or other technical problems. No failures occurred because of the TAMs or the coordinator.

3 Using a group of TAMs to abstract a complex task

In this paper, we call a *complex task* a task that can be decomposed into a collection of atomic tasks, referred to as subtasks. In particular, we consider complex tasks that can be decomposed into a collection of stationary atomic subtasks, each of which can be abstracted by a TAM as presented in Sect. 2. Atomic subtasks might require to be performed in a given order or concurrently. We call the logical and hierarchical structure of the subtasks the *interrelationship* between subtasks. Examples of complex tasks are "harvest an object and store it at a central location", "open a faucet while holding a bucket under it", and "push two buttons at the same time at different locations". Note that complex tasks are not necessarily multi-robot tasks. For example, a complex task might require a single robot to execute several atomic subtasks, one after the other.

Stationary atomic subtasks might be contiguous in space or might take place at different locations. In the latter case, robots might need to travel from one location to the other in order to carry out the complex task. This implies that a complex task that is composed of stationary atomic subtasks could be non-stationary.

In order to use a group of TAMs to abstract a complex task, we first need to model the complex task as a collection of stationary atomic tasks, which are then individually abstracted by a group of TAMs. We propose a two-level approach to modeling complex tasks. The goal of the *high-level model* is to describe the hierarchical structure of a complex task without having to consider all the details of the interrelationships between its subtasks. The model is a convenient high-level description of complex tasks and serves as a basis for classifying and comparing them. The goal of the *low-level model* is to define the details of the interrelationships between the subtasks of a complex task. The model serves as a blueprint for the software that implements these interrelationships between TAMs.

We use well-known visual modeling languages for both levels, more specifically, we use UML 2.x activity diagrams for the high-level model (Rumbaugh et al. 2004) and Petri nets for the low-level model (Petri and Reisig 2008). UML 2.x activity diagrams are appropriate for the high-level model as they are intuitive and convenient to use (Rumbaugh et al. 2004). Petri nets are appropriate for the low-level model because they have a well-defined execution semantics that allows one to simulate them.[5]

3.1 High-level model

In order to model the hierarchical structure of a complex task, we decompose the task into its constituent subtasks. The complexity of the original task resides in the interrelationships between its subtasks. Decomposition is recursive, that is, subtasks can potentially be decomposed further; a subtask of a task can therefore consist of subtasks, as well. Decomposition stops once all decomposable subtasks have been decomposed.

We call the hierarchical structure formed by the subtasks of the complex task the *task relationship graph*. A task relationship graph is a directed acyclic graph: there is a direction

[5] It should be noted that, although the semantics of activity diagrams is loosely based on Petri nets, activity diagrams are unsuitable for simulation because "the rules for activity execution are not clearly explained and defined in the UML specification" (Spiteri Staines 2010).

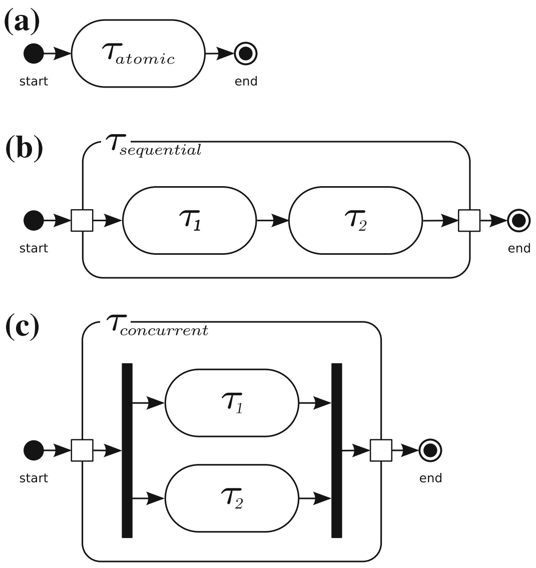


Fig. 6 High-level models of the basic task types, expressed using UML 2.x activity diagrams. **a** An atomic task; **b** A complex task whose subtasks have a sequential interrelationship; **c** A complex task whose subtasks have a concurrent interrelationship

in which the graph has to be traversed in order to execute the original task. Furthermore, each of the nodes of this graph may be a task relationship graph in itself, that is, the graph may be nested.

As defined in Sect. 2, an atomic task is a task that can be performed by a single robot. An atomic task cannot be decomposed, and can be directly abstracted with a TAM. A *complex task*, on the other hand, can be decomposed into several atomic subtasks, each of which can be abstracted with a TAM. The group of TAMs that abstracts the different subtasks forms the abstraction of the complex task. We define a *task instance* as a specific realization of a given task. We describe the task relationship graph visually via UML 2.x activity diagrams (Rumbaugh et al. 2004): we use the UML 2.x activity diagram primitives called *actions* to describe atomic tasks (see Fig. 6a) and *activities* to describe complex tasks (see Fig. 6a, b). We distinguish complex tasks on the basis of their interrelationships:

– A *sequential* interrelationship requires that the subtasks are executed in a given order (see Fig. 6b). An example is a task in which one robot starts to pull a stick from the ground, and that pulling motion must be continued by a second robot (see, e.g., Ijspeert et al. 2001).
– A *concurrent* interrelationship requires that the subtasks are executed at the same time (see Fig. 6c). An example is an area coverage task in which several robots must occupy pre-defined spatially distributed positions in the environment (see, e.g., Berman et al. 2009).

Recursive task decomposition used in conjunction with the described types of task interrelationships yields a powerful yet simple approach to model various complex tasks.

3.2 Low-level model

The high-level model is convenient to use, but omits some of the details of the interrelationships between subtasks. Examples of these details are “two sequential tasks must be executed by the same robot”, “two concurrent tasks have to start or end at the same moment in time”,

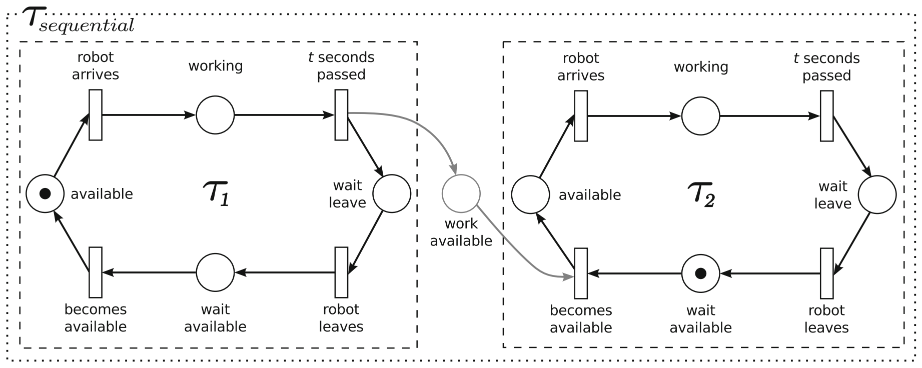


Fig. 7 Low-level model of a complex task $\tau_{\text{sequential}}$ whose subtasks have a sequential interrelationship. The model is expressed using a limited Petri net; all places have a capacity of 1 and all transitions have a weight of 1. Two subnets model the state of the atomic subtasks τ_1 and τ_2, demarcated by *dashed lines*. The place work available represents the sequential interrelationship that links the two subtasks, depicted between the two atomic subtasks

and "two sequential tasks feature a blocking work transfer" (i.e., the first task cannot complete unless the second has started, see Pini et al. 2011; Brutschy et al. 2014).

In order to properly model these details, we propose to use limited capacity Petri nets (Petri and Reisig 2008). Petri nets offer, just as UML 2.x activity diagrams, a graphical notation for stepwise processes that include sequential and concurrent execution. Contrary to activity diagrams, the flow of execution in a Petri net can be simulated, which allows the researcher to test the model before using it in a physical experiment.

We propose to create the Petri net model following a "bottom-up" approach: one starts from the atomic tasks in the high-level model, and iteratively adds the interrelationships that form the encompassing complex tasks until the whole task relationship graph has been modeled.

We model an atomic task as a restricted Petri net called *state machine*. In a state machine, every transition has exactly one pre- and one post-condition, and all markings consist of a single token. We model each atomic task using the same Petri net, which consists of a sequence of states. Transitions between these states are triggered by external events.

We model a complex task on the basis of the models of its atomic subtasks, which form subnets in the Petri net of the complex task. Interrelationships between these atomic subtasks are modeled by adding conditions between state transitions of the atomic tasks. Adding conditions to the state transitions implies that more than one token can be in a given marking. This effectively turns the state machine into a full Petri net whose state is distributed; as a result, the Petri net can model concurrency.

Figure 7 shows a Petri net model of a complex task $\tau_{\text{sequential}}$ that consists of two atomic subtasks with a sequential interrelationship (the matching high-level model is depicted in Fig. 6b). The two atomic subtasks τ_1 and τ_2 are subnets of the overall net (designated by dashed lines in Fig. 7). The sequential interrelationship is such that subtask τ_1 needs to be completed before τ_2 becomes available. We model this interrelationship by adding a place work available to the model, which receives a token each time τ_1 has been completed. We make this place a condition for τ_2 to transition from wait available to available. As a result, τ_2 cannot become available before τ_1 has been completed at least once.

In the example, all places have a capacity of 1 and all transitions have a weight of 1. The capacity of the condition work available models, for example, the capacity of a cache site that stores material transferred from τ_1 to τ_2 (see Pini et al. 2011, 2013 for an example of a task with such a cache site). Accordingly, this capacity could be higher than 1 in order to model a bigger cache site. More complex interrelationships can be modeled by adding more conditions, for example, a blocking interrelationship between τ_1 and τ_2 (see Sect. 4).

3.3 Using the model in experiments

In an experiment, a complex task is abstracted using a group of TAMs. These TAMs are controlled using the control framework described in Sect. 2.3. More specifically, the low-level model of a complex task is implemented on the coordinator that centrally controls all TAMs in the experiment. State changes of this model are triggered by events that happen at the TAMs, and may result in commands that change the behavior of one or multiple TAMs. For example, if a TAM reports to the coordinator that its task has been completed, the model switches state, which in turn causes the TAM in question to switch off and other TAMs to become available. As such, the low-level model serves as a "blueprint" for the software that controls the behavior of the individual TAMs.

Note that the high-level model is not directly required for experimentation other than being a convenient high-level description of a given complex task that serves as a basis for classifying and comparing it.

4 Demonstration

In this section, we demonstrate the use of the TAM. We show how a complex task can be abstracted using several TAMs and how the researcher can leverage the control framework to conduct an experiment.

The task that the robots have to tackle is a complex task that consists of three atomic subtasks. We perform two demonstrations: in the first demonstration, 6 e-puck robots have to tackle a single instance of the complex task; in the second demonstration, 20 e-puck robots have to tackle 6 instances of the complex task.

We assume that the task to be performed by the robots is a disaster response task as it might occur after a nuclear accident. The specific disaster response task τ_{response} is a complex task that consists of two subtasks with a sequential interrelationship: (1) τ_{open}, the task of opening the reactor airlock and (2) τ_{repair}, the task of repairing something inside the reactor. The task τ_{open} requires two robots to act concurrently on the airlock. To this end, each robot executes one of two atomic subtasks, τ_{left} or τ_{right}. After the airlock has been opened, it has to be kept open by the robots until a third robot has entered the reactor chamber. Once the third robot is in the reactor chamber, the robots of τ_{open} can leave. The third robot can then work on the task τ_{repair}. The disaster response task τ_{response} is completed once the reactor has been repaired by completing τ_{repair}. Figure 8 gives a visual representation of its task relationship graph described using UML 2.x activity diagrams.

In order to abstract τ_{response} using a group of TAMs, we have to transform its high-level model into a low-level model based on a Petri net. This low-level model can then be implemented on the coordinator. Figure 9 shows the reduced version of this Petri net.[6] We model each atomic task using a subnet of the same structure: 5 places and 5 transitions model

[6] By convention, the places of a Petri net can be omitted in order to visualize better the structure of the net (Petri and Reisig 2008). The full version of the Petri net and instructions for simulating it can be found in the supplementary online material (Brutschy et al. 2013).

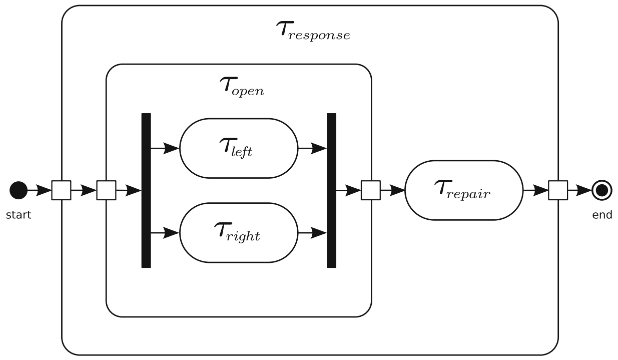


Fig. 8 High-level model of the complex task τ_{response}. Each of the atomic tasks has to be tackled by a single robot

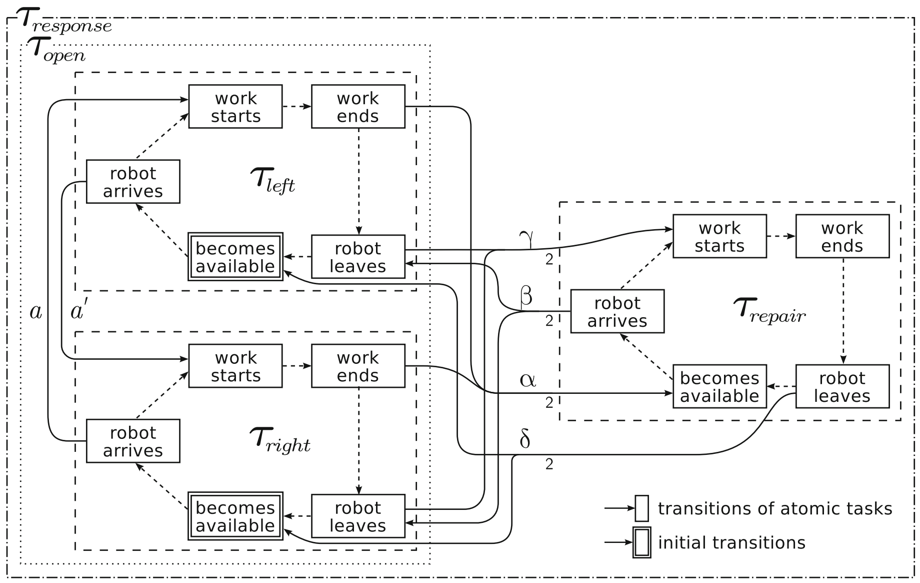
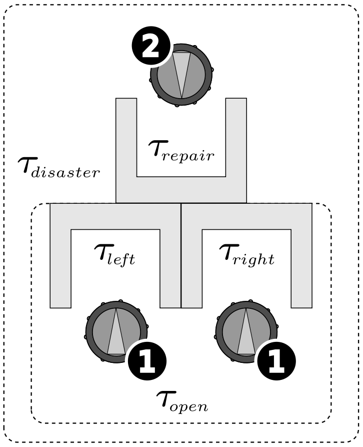
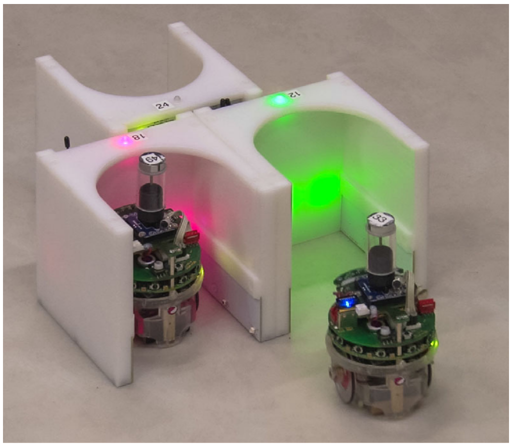


Fig. 9 Low-level Petri net model of the example task τ_{response} (reduced version without places) (see Footnote 6). Please note that, as the initial marking cannot be visualized in the reduced version, we denote transitions that can fire at the start of the demonstration using a double border. The weight of all edges is 1 unless indicated otherwise. Edges that are internal to the functioning of an individual TAM are represented using *dashed lines*. Edges labeled a and a' model the concurrent interrelationship: a robot that is ready to work on τ_{left} can start working once a robot arrives to work on τ_{right} (and vice versa). Edges labeled using Greek letters model the sequential interrelationship: once τ_{left} and τ_{right} have been completed, τ_{repair} becomes available (edge α); a robot that arrives to work on τ_{repair} allows the robots in τ_{right} and τ_{left} to leave (edge β); once these robots have left, work can start on τ_{repair} (edge γ); once work on τ_{repair} finishes and the robot leaves, τ_{right} and τ_{left} can become available anew (edge δ). *Dotted/dashed boxes* indicate task boundaries

the internal state of an atomic task. The transitions of the three atomic tasks τ_{left}, τ_{right}, and τ_{repair} are described in Fig. 9 (labeled using Arabic numerals).

The two atomic tasks τ_{left} and τ_{right} are subtasks of the complex task τ_{open} and have a concurrent interrelationship. This interrelationship is modeled such that work on the tasks





(a) **(b)**

Fig. 10 **a** We use three TAMs to abstract τ_{response}, with each TAM abstracting one of the three atomic subtasks τ_{left}, τ_{right}, and τ_{repair} (see also Fig. 8). *Dotted lines* indicate the complex tasks τ_{open} and τ_{response}. The *white numbers in the black circles* designate the order of execution. **b** Close-up of the TAMs taken during one of the demonstrations. A robot already entered into the TAM that abstracts task τ_{left}. The TAM signals the robot to wait by changing the color of its LEDs to *pink*. The TAM abstracting τ_{right} signals the approaching robot that its associated task is available by its *green* LEDs. The third TAM, abstracting τ_{repair}, is still idle as its sequential interrelationship with the complex task τ_{open} requires that τ_{left} and τ_{right} are completed before τ_{repair} can become available (Color figure online)

can only start when both robots are present, and each robot can leave only after both robots completed their work (edges labeled using Latin letters in Fig. 9).

The complex task τ_{open} and the atomic subtask τ_{repair} have a sequential interrelationship. This interrelationship is such that the robots that completed τ_{open} must wait for a robot to arrive for τ_{repair}. Furthermore, the arriving robot has to wait for the others to leave before it can start to work on τ_{repair} (edges labeled using Greek letters in Fig. 9).

As τ_{response} consists of, in total, three atomic subtasks, we use three TAMs to abstract a single instance of τ_{response}, with each TAM abstracting one of the three atomic subtasks. We use the control framework described in Sect. 2.3 to implement the low-level model. Note that in the following we use the term "TAM τ_x" interchangeably with the term "task τ_x". Fig. 10a illustrates how a single instance of τ_{response} is abstracted using three TAMs.

We use e-puck robots in the configuration shown on the right side of Fig. 2. All robots use an instance of the same controller: by default, robots perform a random walk. If a robot perceives a TAM using its camera, it tries to enter into it in order to start working on the associated task. The robots follow a simple greedy strategy to select tasks, that is, every robot tries to work on any available task it encounters. Upon completion of the task, the robot leaves the TAM and starts again to perform a random walk.

Both demonstrations have been recorded using an overhead camera. Additionally, we recorded all the available data from the TAMs using the coordinator. Videos and data are available in the supplementary online material (Brutschy et al. 2013).

4.1 Single-instance demonstration

The first demonstration illustrates how the design of the control framework can be leveraged to collect detailed data from each TAM.

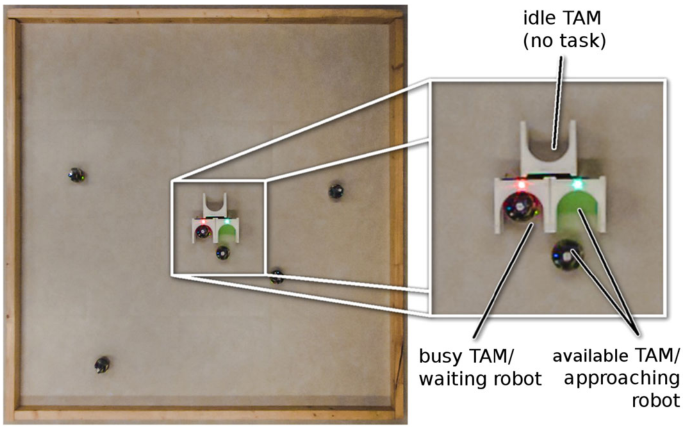


Fig. 11 Snapshot of the single-instance demonstration, taken with an overhead camera in the same situation as shown in Fig. 10a. The arena is a 4 m^2 square. A single instance of the task τ_{response}, abstracted using three TAMs, is placed in the center of the arena. We use 6 e-puck robots

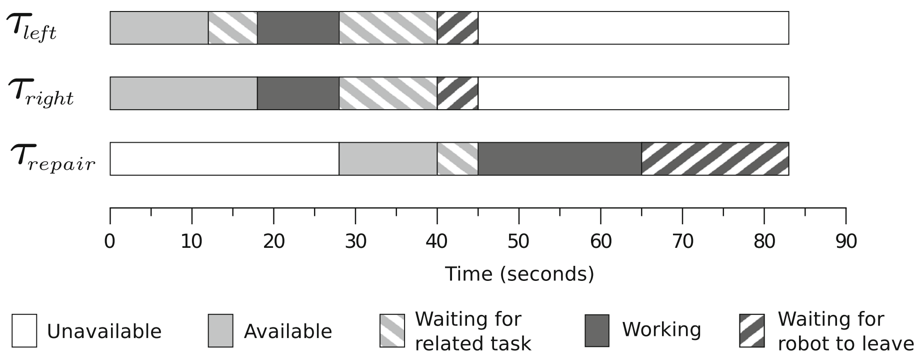


Fig. 12 Evolution of the task τ_{response} over time in terms of state changes of its subtasks. The times shown are the result of an exemplary execution of a single instance of τ_{response}

We use a 4 m^2 square arena. The three TAMs are configured as shown in Fig. 10b and placed in the center of the arena. Figure 11 shows a snapshot that illustrates the arena and the position of the TAMs. At the beginning of the demonstration, 6 e-puck robots are randomly positioned in the arena. We set the duration of τ_{left} and τ_{right} to 10 s and of τ_{repair} to 20 s. The demonstration terminates as soon as task τ_{response} has been completed once. Figure 12 illustrates how τ_{response} evolves over time.

The coordinator implements the model of the task, and governs the state of each TAM. Additionally, each TAM reports all events and changes in sensory data to the coordinator, which facilitates data collection during an experiment. The ability to record task-related data enables the study of algorithms that leverage this kind of data. For example, we can collect various task-related metrics such as the time it takes until a robot arrives to work on a task, or the time a robot working on a task has to wait for a partner—data that are not trivial to obtain when using ad hoc solutions for task abstraction. Furthermore, the TAM (and thereby the coordinator) can record the identity of each robot.

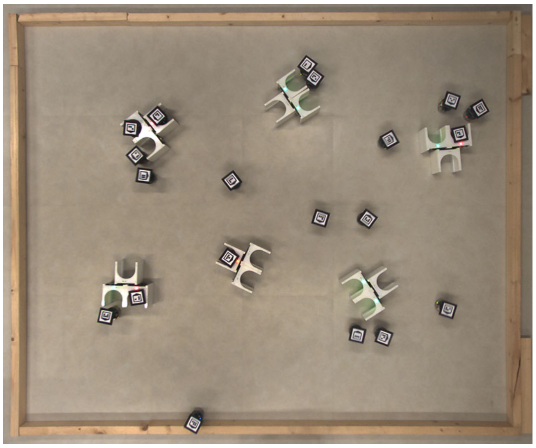


Fig. 13 Snapshot of the multi-instance demonstration, taken with an overhead camera. The arena has dimensions of 2.7 m×2.2 m. The six task instances have been placed at random locations in the arena. We use a swarm of 20 e-puck robots. The *black-and-white tags* on top of the robots are used for tracking the robots using a ceiling-mounted tracking system (Stranieri et al. 2013)

The demonstration illustrates how the TAM broadens the range of tasks that can be studied in an experiment. First, any task-related time can be closely controlled and modified during an experiment. For example, the object transport task studied by Pini et al. (2011) can be completed by the robots in two ways: individually by traveling through a corridor or collectively by partitioning the task and exchanging objects at a cache site. By abstracting the cache site using a set of TAMs, Pini et al. could vary the advantage of using the cache site over using the corridor. Without the TAM, this kind of study would require modifying the length of the corridor, which might incur unintended changes in the environmental parameters (e.g., the robot density would change).

Second, the capability of identifying robots enables tasks that are specific to robots. An example is the task studied by Brutschy et al. (2012): robots specialize in one of two possible task types. As each robot has its individual level of specialization, the duration of a task might be different for each robot. A study of this type would not be possible without the capability of the TAM of communicating with the robots.

4.2 Multi-instance demonstration

Extending upon the first demonstration, the second demonstration illustrates how the researcher can use the TAM to conduct experiments involving larger swarms and several instances of complex tasks. Again, we use the control framework to record detailed data such as the number of successful task executions per atomic subtask. In this demonstration, we use 18 TAMs and 20 e-puck robots.

We use a rectangular arena with dimensions of 2.7 m×2.2 m. Six instances of τ_{response} are placed at random locations in the arena. At the beginning of the demonstration, 20 e-puck robots are randomly positioned in the arena. Again, we set the duration of τ_{left} and τ_{right} to 10 s and of τ_{repair} to 20 s. The demonstration terminates after 5 min. Figure 13 shows a snapshot that illustrates the setup of the arena and the position of the TAMs.

In terms of data collection, we record the same data as in the first demonstration, but for many robots and several task instances, thereby illustrating how the researcher can record large amounts of data over several task instances and/or experiments. The demonstration also illustrates how the centralized recording of data allows researchers to correlate it from the TAMs with the data of an external tracking system—if necessary, in real time.

In terms of experimental setup, the demonstration illustrates how to deploy several instances of the same complex task, and how the central coordinator can control a large number of TAMs in parallel, using the scalable mesh network to communicate with the TAMs.

5 Tasks studied in the literature

In this section, we review the literature with respect to the tasks studied. We describe the task studied by each work using the high-level model presented in Sect. 3, and group works according to similarities in their high-level model.

This review serves two purposes. First, it allows us to substantiate the claim we made in Sect. 1: the use of ad hoc solutions for task abstraction limits the complexity of the tasks that can be studied. As we will show, the hierarchical structure of the majority of tasks studied in the swarm robotics literature is relatively simple: tasks are either atomic or consist of a single complex task. However, real-world tasks commonly exhibit a higher complexity than this. Second, by modeling each task using the high-level model, we demonstrate how to use this model to abstract various complex tasks. This, in turn, outlines how these tasks could be abstracted using the TAM. Note that we refrain from detailing how to use the TAM for each work; see Sect. 4 for an example of how to pass from the high-level model of a task to abstracting it using the TAM.

Please note that we focus on works in which real robots perform atomic or complex tasks that can be represented using the TAM. This excludes for example many spatially organizing behaviors and navigation behaviors.

5.1 Atomic tasks

There are several works that study tasks that are atomic. Atomic tasks cannot be decomposed into subtasks and can be readily abstracted using a single TAM. See Fig. 6a for the high-level model of an atomic task.

Matarić et al. (2003) studied emergency handling by a group of robots. Emergencies are atomic tasks that appear in the environment and have to be attended by a single robot. Brutschy et al. (2012) studied atomic tasks with the additional requirement that the robots must individually specialize on one type of task, while adhering to an optimal allocation at the level of the swarm.

5.2 Complex tasks consisting only of atomic subtasks

Complex tasks that consist exclusively of atomic subtasks are commonly studied in the literature. As mentioned in Sect. 3, we distinguish these tasks according to the interrelationship between their subtasks. Complex tasks whose atomic subtasks have the same type of interrelationship differ only in the number of these subtasks. Note that, while the exemplary high-level models referenced in the following have two subtasks, they can be easily extended to a higher number of subtasks.

A complex task whose subtasks have a sequential interrelationship requires that subtasks are executed in a given order—see Fig. 6b for the high-level model of such a task. A commonly studied problem that involves complex tasks of this type is foraging for food or energy. In this problem, robots must balance energy consumed by the process of foraging with the energy provided by the collected food items (Krieger and Billeter 2000; Li et al. 2004; Labella et al. 2006). The subtasks of the foraging task exhibit a sequential interrelationship which lies in

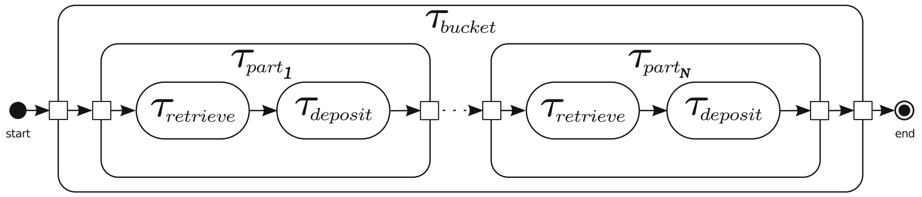


Fig. 14 High-level UML model of bucket-brigading tasks: the overall task τ_{bucket} of transporting an object is partitioned into a sequence of N complex tasks. Each complex task consists of two atomic subtasks for retrieving and depositing the object

the fact that robots first have to collect an item in the environment and then transport it to a predefined drop-off location. The same type of complex task has been studied in the form of a "waste cleanup" scenario (see, e.g., Parker 1998). Please note that in this simple version of the foraging problem, robots do not need to collaborate in order to complete a single task instance. Ijspeert et al. (2001) studied a task with a sequential interrelationship. The goal of the robots is to pull sticks from the ground. The length of the sticks is such that a robot cannot pull it from the ground in a single motion; instead, a second robot has to continue the pulling motion in order to complete the task.

A complex task whose subtasks have a concurrent interrelationship requires that subtasks are executed at the same time—see Fig. 6c for the high-level model of such a task. Commonly studied complex tasks with this type of interrelationship among their atomic subtasks are collective transport tasks (Donald et al. 1997; Kube and Bonabeau 2000; Groß and Dorigo 2009). Typically, all robots execute the same action, that is, the subtasks have a concurrent interrelationship.

5.3 Nested complex tasks

Nested complex tasks are tasks that have some subtasks that are complex tasks as well. An example of such a task is the disaster response task presented in Sect. 4. Note that, as tasks can be arbitrarily nested, the high-level model of such a task can have various shapes.

The majority of the works that study nested complex tasks consider the same type of task: "bucket brigading". Bucket brigading is a special case of a foraging task: robots divide a transportation task over a longer distance into multiple smaller subtasks. Each subtask consists of transporting an object for a limited distance and subsequently transferring it to a robot working on the next subtask. Hence, the overall task is a sequence of foraging tasks, and each foraging task consists of a sequence of two atomic tasks. Figure 14 shows the high-level model of a bucket-brigading task. Many works study complex tasks in the form of bucket brigading. Most works use fixed partition sizes (Fontan and Matarić 1996; Goldberg and Matarić 2002). Brutschy et al. (2014) studied self-organized allocation to such tasks with two partitions of fixed size. The work published by Pini et al. (2014) is, to the best of our knowledge, the only one that studied tasks with sequential interrelationships and dynamic partition sizes.

Complex tasks that consist of multiple levels of nested complex tasks are rarely studied in the literature, most likely due to the complexity and cost of the experiments needed to study them. In the following, we outline two works that study such a task; we refer the reader to the respective publication for details on each subtask. In the context of Swarm-bots project,[7] Nouyan et al. (2009) studied task allocation in a collective transportation task—see Fig. 15a

[7] http://www.swarm-bots.org/.

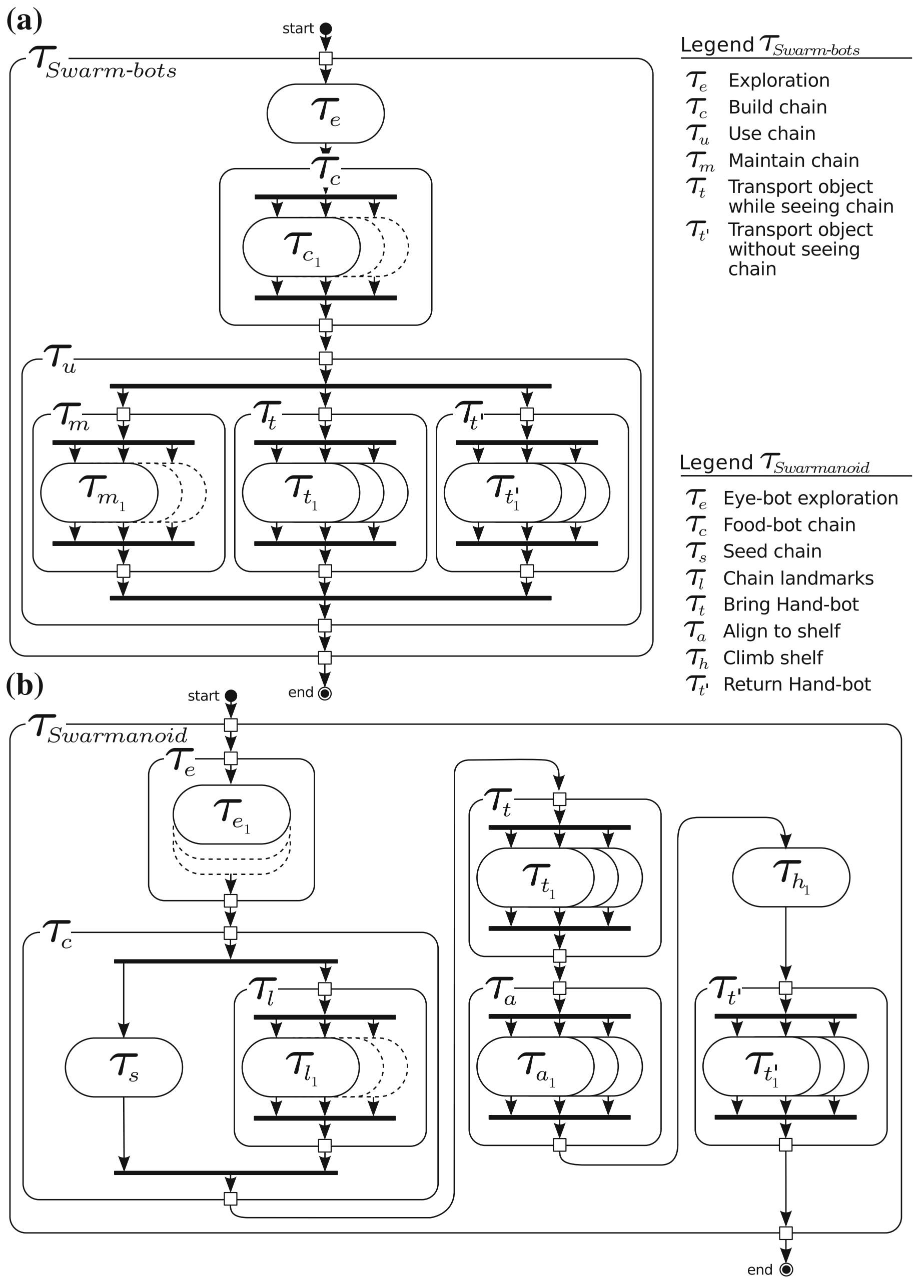

(a)
start
$\mathcal{T}_{Swarm\text{-}bots}$
$\mathcal{T}_e$
$\mathcal{T}_c$
$\mathcal{T}_{c_1}$
$\mathcal{T}_u$
$\mathcal{T}_m$
$\mathcal{T}_{m_1}$
$\mathcal{T}_t$
$\mathcal{T}_{t_1}$
$\mathcal{T}_{t'}$
$\mathcal{T}_{t'_1}$
end
Legend $\mathcal{T}_{Swarm\text{-}bots}$
$\mathcal{T}_e$ Exploration
$\mathcal{T}_c$ Build chain
$\mathcal{T}_u$ Use chain
$\mathcal{T}_m$ Maintain chain
$\mathcal{T}_t$ Transport object while seeing chain
$\mathcal{T}_{t'}$ Transport object without seeing chain
(b)
start
$\mathcal{T}_{Swarmanoid}$
$\mathcal{T}_e$
$\mathcal{T}_{e_1}$
$\mathcal{T}_c$
$\mathcal{T}_s$
$\mathcal{T}_l$
$\mathcal{T}_{l_1}$
$\mathcal{T}_t$
$\mathcal{T}_{t_1}$
$\mathcal{T}_a$
$\mathcal{T}_{a_1}$
$\mathcal{T}_{h_1}$
$\mathcal{T}_{t'}$
$\mathcal{T}_{t'_1}$
end
Legend $\mathcal{T}_{Swarmanoid}$
$\mathcal{T}_e$ Eye-bot exploration
$\mathcal{T}_c$ Food-bot chain
$\mathcal{T}_s$ Seed chain
$\mathcal{T}_l$ Chain landmarks
$\mathcal{T}_t$ Bring Hand-bot
$\mathcal{T}_a$ Align to shelf
$\mathcal{T}_h$ Climb shelf
$\mathcal{T}_{t'}$ Return Hand-bot

Fig. 15 High-level UML model of two tasks with multiple levels of nested complex tasks: **a** Task studied by Nouyan et al. (2009) in the context of the Swarm-bots project; **b** Task studied by the Swarmanoid project (Dorigo et al. 2013). Atomic tasks with *dashed lines* represent tasks where the number of tasks on the same level is more than three or can vary

for the high-level model of this task. The complexity of the task lies in the fact that robots first establish a chain of landmarks between source and nest, which is subsequently used by other robots to navigate while collectively transporting an object from the source to the nest.

One of the most complex tasks found in the swarm robotics literature has been presented by the Swarmanoid project:[8] a swarm collectively explores an environment, identifies an object to retrieve, and uses self-assembly and collective transport to retrieve it (Dorigo et al. 2013)—see Fig. 15b for the high-level model of this task.[9]

5.4 Discussion

The review of the literature shows that the vast majority of works consider tasks without or with a limited number of interrelated subtasks. Moreover, to the best of our knowledge, most of the works that consider nested complex tasks tackle tasks that exhibit only one type of interrelationships, namely sequential bucket-brigading tasks. We speculate that the restriction to tasks with a low number of interrelated subtasks is due to the costs involved in studying such tasks: performing real-robot experiments for these tasks requires considerable effort and resources, as illustrated by the two studies that consider multiple levels of nested complex tasks. As a result, swarm robotics systems are to date incapable of tackling complex tasks that consist of a large number of interrelated subtasks; a fact that limits the possible application of swarm robotics to real-world problems.

From this insight, two directions for future research are discernible: one, the development of group behaviors and collective decision processes that allow a swarm to tackle tasks of this kind, and two, the development of robotics hardware that is sufficiently capable, cost-effective, and robust to apply a swarm of robots to such tasks. The TAM enables researchers to work towards the first direction.

6 Conclusions

In this paper, we proposed a new approach to abstract complex tasks in swarm robotics research. This approach is based on a novel physical device, called the TAM. The purpose of the TAM is to abstract stationary single-robot tasks to be performed by the e-puck robot.

Complex multi-robot tasks, on the other hand, have to be modeled as a collection of single-robot subtasks before they can be abstracted using a group of TAMs. We presented a collection of tools for modeling complex tasks as their constituent single-robot subtasks and the interrelationships between them. Additionally to the modeling tools, we presented a framework that allows researchers to control large groups of TAMs and implement the interrelationships identified by the model. As a result, all complex tasks that can be modeled with the proposed tools can be abstracted using groups of TAMs.

The TAM, used together with the proposed modeling tools and control framework, enables research on cooperative behaviors for complex tasks using large swarms of robots. Furthermore, experiments can be conducted using simple, cost-effective robots such as the e-puck. These experiments would otherwise require costly solutions such as specialized robots or ad hoc task abstractions.

For demonstration purposes, we abstracted an example task, first by modeling it at a high level, then at a low level, and finally by conducting two demonstrations involving e-

[8] http://www.swarmanoid.org/.

[9] See http://youtu.be/M2nn1X9Xlps for a movie describing the swarm and its task.

puck robots and several TAMs. The demonstrations illustrate how the TAM can be used to abstract complex multi-robot tasks in a real-robot experiment, and how the proposed control framework can facilitate the researcher's job of conducting experiments.

We reviewed the swarm robotics literature with respect to the tasks studied. Our review shows that only a few works study tasks with a large number of interrelated subtasks—a limitation that, as we speculate, is due to the ad hoc fashion in which most works abstract tasks. We are confident that the TAM will allow the research community to advance swarm robotics beyond tasks of this limited complexity.

Even though we developed the TAM for the e-puck robot, the design can be easily adapted to other robotic platforms. To this end, we released all the components of the TAM as open-source, thereby allowing other research groups to adapt the TAM to their research.

The variety of tasks that can be studied using the TAM is documented by the various works that rely on it for task abstraction (Pini et al. 2011, 2013; Brutschy et al. 2012; Castillo-Cagigal et al. 2014; Brambilla et al. 2014). Immediate future work will be targeted at replicating some of these experiments with larger swarms of e-pucks using up to 50 TAMs.

Acknowledgments The authors would like to thank Álvaro Gutiérrez and Manuel Castillo-Cagigal for their help with designing the electronics of the TAM. The research leading to the results presented in this paper has received funding through the ERC Advanced Grant "E-SWARM: Engineering Swarm Intelligence Systems" (ERC Grant Agreement No. 246939). Arne Brutschy, Manuele Brambilla, Marco Dorigo, and Mauro Birattari acknowledge support from the Belgian F.R.S.—FNRS of Belgium's Wallonia-Brussels Federation.

References

Acerbi, A., Marocco, D., & Nolfi, S. (2007). Social facilitation on the development of foraging behaviors in a population of autonomous robots. In F. Almeida e Costa, L. Rocha, E. Costa, I. Harvey, & A. Coutinho (Eds.), *Advances in artificial life* (Vol. 4648, pp. 625–634)., Lecture notes in computer science Berlin: Springer.

Banzi, M. (2008). *Getting started with Arduino*. Sebastopol, CA: O'Reilly Media.

Beni, G. (2005). From swarm intelligence to swarm robotics. In E. Şahin & W. M. Spears (Eds.), *Swarm robotics* (Vol. 3342, pp. 1–9)., Lecture notes in computer science Berlin: Springer.

Berman, S., Halász, A., Hsieh, M. A., & Kumar, V. (2009). Optimized stochastic policies for task allocation in swarms of robots. *IEEE Transactions on Robotics*, *25*(4), 927–937.

Brambilla, M., Brutschy, A., Dorigo, M., & Birattari, M. (2014). Property-driven design for robot swarms: A design method based on prescriptive modeling and model checking. *ACM Transactions on Autonomous and Adaptive Systems*, *9*(4), 17:1–17:28.

Brambilla, M., Ferrante, E., Birattari, M., & Dorigo, M. (2013). Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, *7*(1), 1–41.

Brutschy, A. (2014). *The TAM: A device for task abstraction in swarm robotics research*. Technical Report TR/IRIDIA/2010-015.005, Belgium: IRIDIA, Université Libre de Bruxelles.

Brutschy, A., Garattoni, L., Brambilla, M., Francesca, G., Pini, G., Dorigo, M., & Birattari, M. (2013). *The TAM: Abstracting complex tasks in swarm robotics research—supplementary online material*. Retrieved from http://iridia.ulb.ac.be/supp/IridiaSupp2012-002/.

Brutschy, A., Pini, G., Pinciroli, C., Birattari, M., & Dorigo, M. (2014). Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous Agents and Multi-Agent Systems*, *28*(1), 101–125.

Brutschy, A., Tran, N.-L., Baiboun, N., Frison, M., Pini, G., Roli, A., et al. (2012). Costs and benefits of behavioral specialization. *Robotics and Autonomous Systems*, *60*(11), 1408–1420.

Campo, A., Garnier, S., Dédriche, O., Zekkri, M., & Dorigo, M. (2011). Self-organized discrimination of resources. *PLoS One*, *6*(5), e19888.

Castillo-Cagigal, M., Brutschy, A., Gutiérrez, Á., & Birattari, M. (2014). Temporal task allocation in periodic environments: An approach based on synchronization (Vol. 8667). In*Proceedings of the 9th International Conference on Swarm Intelligence (ANTS'14)* (pp. 182–193). Lecture Notes in Computer Science Berlin/Heidelberg, Germany: Springer.

Donald, B. R., Jennings, J., & Rus, D. (1997). Information invariants for distributed manipulation. *The International Journal of Robotics Research*, *16*(5), 673–702.

Dorigo, M., Birattari, M., & Brambilla, M. (2014). Swarm robotics. *Scholarpedia*, *9*(1), 1463.

Dorigo, M., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., Baaboura, T., et al. (2013). Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, *20*(4), 60–71.

Fontan, M. S., & Matarić, M. J. (1996). A study of territoriality: The role of critical mass in adaptive task division. In P. Maes, M. J. Matarić, J.-A. Meyer, J. Pollack, & S. Wilson (Eds.), *From animals to animats 4: Proceedings of the Fourth International Conference of Simulation of Adaptive Behavior* (pp. 553–561). Cambridge, MA: MIT Press.

Francesca, G., Brambilla, M., Brutschy, A., Garattoni, L., Miletitch, R., Podevijn, G., et al. (2014a). An experiment in automatic design of robot swarms: AutoMoDe-Vanilla, EvoStick, and human experts (Vol. 8667). In M. Dorigo, M. Birattari, S. Garnier, H. H. M. M. de Oca, C. Solnon, & T. Stützle (Eds.), *Proceedings of the 9th International Conference on Swarm Intelligence (ANTS'14)* (pp. 25–37). Lecture Notes in Computer Science, Springer: Berlin/Heidelberg, Germany.

Francesca, G., Brambilla, M., Brutschy, A., Trianni, V., & Birattari, M. (2014b). AutoMoDe: A novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*, *8*(2), 89–112.

Gauci, M., Chen, J., Li, W., Dodd, T. J., & Groß, R. (2014). Self-organized aggregation without computation. *The International Journal of Robotics Research*, *33*(8), 1145–1161.

Goldberg, D., & Matarić, M. J. (2002). Design and evaluation of robust behavior-based controllers. In T. Balch & L. E. Parker (Eds.), *Robot teams: from diversity to polymorphism* (pp. 315–344). Natick, MA: A. K. Peters.

Groß, R., & Dorigo, M. (2009). Towards group transport by swarms of robots. *International Journal of Bio-Inspired Computation*, *1*(1–2), 1–13.

Gutiérrez, A., Campo, A., Dorigo, M., Amor, D., Magdalena, L., & Monasterio-Huelin, F. (2008). An open localisation and local communication embodied sensor. *Sensors*, *11*(8), 7545–7563.

Ijspeert, A. J., Martinoli, A., Billard, A., & Gambardella, L. M. (2001). Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots*, *11*(2), 149–171.

Jakobi, N., Husbands, A., & P., & A. Harvey, I., (1995). Noise and the reality gap: The use of simulation in evolutionary robotics (Vol. 929). In F. Morán, A. Moreno, J. J. Merelo, & P. Chacón (Eds.), *Swarm Robotics* (pp. 704–720). Advances in Artificial Life, Springer: Berlin/Heidelberg, Germany.

Kernbach, S., Nepomnyashchikh, V., Kancheva, T., & Kernbach, O. (2012). Specialization and generalization of robotic behavior in swarm energy foraging. *Mathematical and Computer Modelling of Dynamical Systems*, *18*, 131–152.

Krieger, M. J. B., & Billeter, J.-B. (2000). The call of duty: Self-organized task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, *30*(1–2), 65–84.

Kube, C., & Bonabeau, E. (2000). Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, *30*(1–2), 85–101.

Labella, T. H., Dorigo, M., & Deneubourg, J.-L. (2006). Division of labour in a group of robots inspired by ants' foraging behaviour. *ACM Transactions on Autonomous and Adaptive Systems*, *1*(1), 4–25.

Li, L., Martinoli, A., & Abu-Mostafa, Y. S. (2004). Learning and measuring specialization in collaborative swarm systems. *Adaptive Behavior*, *12*(3–4), 199–212.

Matarić, M. J., Sukhatme, G. S., & Østergaard, E. H. (2003). Multi-robot task allocation in uncertain environments. *Autonomous Robots*, *14*, 255–263.

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., et al. (2009). The e-puck, a robot designed for education in engineering. In P. J. S. Gonçalves, et al. (Eds.), *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions* (pp. 59–65). IPCB: Instituto Politècnico de Castelo Branco, Portugal.

Nouyan, S., Groß, R., Bonani, M., Mondada, F., & Dorigo, M. (2009). Teamwork in self-organized robot colonies. *IEEE Transactions on Evolutionary Computation*, *13*(4), 695–711.

Parker, L. E. (1998). Alliance: An architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, *14*, 220–240.

Petri, C. A., & Reisig, W. (2008). Petri net. *Scholarpedia*, *3*(4), 6477.

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., et al. (2012). ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, *6*(4), 271–295.

Pini, G., Brutschy, A., Frison, M., Roli, A., Birattari, M., & Dorigo, M. (2011). Task partitioning in swarms of robots: An adaptive method for strategy selection. *Swarm Intelligence*, *5*(3–4), 283–304.

Pini, G., Brutschy, A., Scheidler, A., Dorigo, M., & Birattari, M. (2014). Task partitioning in a robot swarm: Retrieving objects by transferring them directly between sequential sub-tasks. *Artificial Life*, *20*(3), 291–317.

Pini, G., Gagliolo, M., Brutschy, A., Dorigo, M., & Birattari, M. (2013). Task partitioning in a robot swarm: A study on the effect of communication. *Swarm Intelligence*, *7*(2–3), 173–199.

Rumbaugh, J., Jacobson, I., & Booch, G. (2004). *The unified modeling language reference manual* (2nd ed.). Upper Saddle River, NJ: Pearson Higher Education.

Sperati, V., Trianni, V., & Nolfi, S. (2008). Evolving coordinated group behaviours through maximisation of mean mutual information. *Swarm Intelligence*, *2*(2), 73–95.

Spiteri Staines, A. (2010). Petri nets applications. In Intuitive transformation of UML2 activities into fundamental modeling concept petri nets and colored petri nets (pp. 673–694). Rijeka, Croatia: InTech Europe.

Stranieri, A., Turgut, A., Francesca, G., Reina, A., Dorigo, M., & Birattari, M. (2013). *IRIDIA's arena tracking system*. Technical Report TR/IRIDIA/2013-013, Belgium: IRIDIA, Université Libre de Bruxelles.