UNIVERSITÄT AUGSBURG
Universitätsbibliothek

OPEN ACCESS
OPUS AUGSBURG

Multipotent systems: a new paradigm for multi-robot applications

Oliver Kosak

Angaben zur Veröffentlichung / Publication details:

Kosak, Oliver. 2018. "Multipotent systems: a new paradigm for multi-robot applications." In *Organic Computing: Doctoral Dissertation Colloquium 2018*, edited by Sven Tomforde and Bernhard Sick, 125–42. Kassel: kassel university press. https://doi.org/10.19211/KUP9783737606974.

Nutzungsbedingungen / Terms of use:

licgercopyright

Dieses Dokument wird unter folgenden Bedingungen zur Verfügung gestellt: / This document is made available under these conditions:
Deutsches Urheberrecht
Weitere Informationen finden Sie unter: / For more information see:
https://www.uni-augsburg.de/de/organisation/bibliothek/publizieren-zitieren-archivieren/publiz/

10

Multipotent Systems: A New Paradigm for Multi-Robot Applications

Oliver Kosak

Institute for Software & Systems Engineering, University of Augsburg, Augsburg, Germany
kosak@isse.de
Home Page: https://www.isse.uni-augsburg.de/

Abstract. Multi-robot applications typically are designed as either homogeneous or heterogeneous systems concerning their software and hardware design. While technologies developed for homogeneous systems provide robustness against failures, they lack versatility concerning applicability. For heterogeneous systems, this observations is true vice versa: While they can be applied to versatile use cases, failures are hard to compensate. We propose the class of *multipotent systems* to merge the benefits of heterogeneous and homogeneous robot systems. For achieving this, we equip an ensemble consisting of mobile robots with self-organisation abilities enabling it to autonomously adapt itself on coalition level as well as on capability level.

Keywords: Multipotent Systems, Multi-agent, Multi-robot, Self-awareness, Self-adaptation, Self-organisation, Organic Computing, Reconfiguration

10.1 Motivation

Yet another multi-robot programming framework, are you serious? The last decade has seen a multitude of different applications and frameworks all having the purpose to deploy multi-robot systems for versatile use cases. The spectrum ranges from search and rescue applications [38, 8, 43, 9] over such used for distributed surveillance of critical infrastructure [34, 35, 52] to those dedicated to environmental research [51, 12, 46]. So why is another framework necessary? The sheer number of different applications, robot designs, and programming frameworks shows: For each new use case the wheel is reinvented over and over again, new robots with new capabilities for new applications have to be invented, designed, built, and programmed with individualised software. We see, requirements concerning the capabilities of robots (e.g., measure certain parameters or interact with the environment) typically change in different use cases, but also in the course of operating

in a specific one. With the approach proposed in this paper, we aim at overcoming this need for specialisation. By separating the concept of capabilities from that of robots in addition to equipping the system with *self-adaptation (SA)* and *self-organisation (SO)* abilities, we enable multi-robot systems to be adapted *before* and autonomously adapt to changing requirements *at* runtime, following the specification paradigm [20] as known from the Organic Computing domain [47, 37]. We use the terms self-adaptation and self-organisation following the definition of [1]:

> *Self-adaptive* systems work in a top-down manner. They evaluate their own global behaviour and change it when the evaluation indicates that they are not accomplishing what they were intended to do, or when better functionality or performance is possible. Such systems typically operate with an explicit internal representation of themselves and their global goals.
>
> *Self-organising* systems work bottom-up. They are composed of a large number of components that interact according to simple and local rules. The global behaviour of the system emerges from these local interactions, and it is difficult to deduce properties of the global system by studying only the local properties of its parts. Such systems do not use internal representations of global properties or goals; they are often inspired by biological or sociological phenomena.

To demonstrate the necessity as well as to evaluate the functionality of our framework, we apply our techniques to the case study of *ScORe missions* [32]. ScORe missions are structured in tasks, where an autonomous system (denominated as **ensemble** in the following) needs to cooperatively **S**earch for certain parameters, **c**ontiniously **O**bserve those, and **Re**act to critical situations. As running example for such a ScORe mission, the scenario of a chemical accident involving a gas leak is illustrated in Fig. 10.1. Repeatedly, such accidents immensely impair peoples' daily life (e.g., accident at BASF in Germany, 2016 or at Arkema in Texas, 2017). During the initial **S**earch task, the ensemble has to cooperatively determine the most harmful gas type as well as its source. In the subsequent **c**ontinuous **O**bservation task, the dissemination of the detected gas has to be tracked while also critical infrastructure (e.g., hospitals) needs to be monitored by the ensemble. If the gas reaches a critical area, in a **Re**act task the ensemble has to alert endangered civilians. Like Fig. 10.1 illustrates, in each task other capabilities are required which have to be provided by the robots participating in the task. For fulfilling these requirements, current approaches [43, 9, 34, 12] introduce appropriately configured robots (concerning hardware and software) for each task. This strategy has multiple drawbacks:

D1) Resource and engineering overhead: The amount of needed robots grows proportionally with the amount of different tasks and use cases.

D2) Tradeoff between robustness or versatility: On the one hand, within heterogeneous systems (consisting of robots each with individual hardware and software), the heterogeneity in robots leads to reduced robustness against failures, i.e., if a specialised robot breaks down, it has to be replaced by an equivalently configured one.

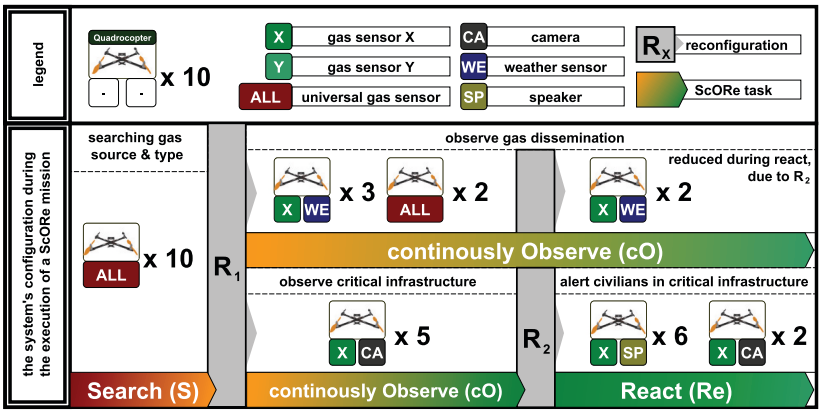


Fig. 10.1: Example of a ScORe mission. The upper part serves as a legend. The lower part shows possible ensemble configurations and task assignments in the course of that mission (dependencies ordered from left to right, including parallel execution) [25].

Homogeneous systems (consisting of robots equal to each other concerning their hardware and software) on the other hand are specialists in solving one dedicated task and can not be used within others.

D3) Planning and allocation complexity: Creating plans that define the procedure of execution for tasks is NP-complete [13] while allocating those tasks to agents is np-hard worsened through "larger team sizes and greater heterogeneity of robots and tasks" [16].

To exemplify how we want to overcome these drawbacks, we concretise our objectives in Section 10.2. Achieving these objectives, multiple technologies have to be developed and integrated with already existing ones. In this paper, we focus on the software architecture necessary for supporting all of them. In Section 10.3, we illustrate how our concept of a layered architecture supports achieving these objectives and how the needed technologies and algorithms are situated within different layers. Thereby, we constitute how we delimit our approach from those of others. In Section 10.6, we give an overview of results we found so far as well as an outlook for the next topics we will investigate in the near future.

10.2 Objectives

The objective of our work is threefold as we want to overcome the three main drawbacks of current approaches introduced in Section 10.1. To achieve this we introduce the new system class of *multipotent systems*, merging the benefits from current homogeneous and heterogeneous systems.

The term *multipotency* is an analogy from biology which fits very smoothly into the context, as multipotent progenitor cells are able to adapt their functionality when brought into new environments [18]. The new system class of multipotent systems is characterised by its homogeneity concerning the software running on each of the system's participants (homogeneity at design time) and its adaptable heterogeneity concerning their provided capabilities (heterogeneity at runtime). By this, benefits from both kinds of systems are inherited. On the one hand, with their *run time heterogeneity*, agents of multipotent systems can be configured very individually to reproduce the versatility of heterogeneous systems [31]. On the other hand, with their *design time homogeneity,* agents in multipotent systems easily can compensate for each other in case of failures as it can be done in homogeneous systems. These characteristics are exploited to tackle *D1 – D3*:

For *D1*, we exploit the ability to reconfigure the ensemble's robots in terms of their capabilities. In comparison to related literature on autonomous, homogeneous or heterogeneous multi-robot systems [14, 46, 33, 10], we do not need to provide appropriately configured robots for each of these tasks but enable the system to adapt to changing requirements in reconfiguration phases (cf. R_1, R_2, R_3 in Fig. 10.1). This approach reduces the needed amount of robots. In our example in Fig. 10.1, a conventional heterogeneous system would need 24 devices (10 devices in **S**, $3+5$ additional differently configured devices in **cO**, and further 6 differently configured devices in **Re**). With our approach exploiting the reconfiguration capabilities of robots, we only need 10 devices that can be configured appropriately in each phase.

To tackle *D2* by increasing robustness, we propose appropriate SA-SO abilities to compensate for failures of different kinds which cannot be planned for at design time (e.g., break down of robots or sensors). As homogeneous systems are robust against interferences per definition [4], this characteristic can also be proposed for multipotent systems. To achieve an *autonomous-as-possible* execution of user-defined tasks, we need to further endow our robots with SA-SO abilities which reduces the number of needed interactions between user and ensemble to a minimum. To enable versatility of the system, we propose SA-SO mechanisms such as coalition formation [45] or resource allocation [7] to adapt to unforeseen situations (e.g., changed environmental conditions).

To address *D3*, we reduce the problem of multi-agent planning to that of single-agent planning [28] by considering the ensemble as one entity (the collective acts as a single agent). Thereby, we map the problem of *multi-robot task allocation* (MRTA) [16] from the class of *multi-robot tasks, multi-task robots* (MR-MT) to the class of *single-robot tasks, multi-task robots* (SR-MT) [16]. This is possible in multipotent systems when collective algorithms originally designed for homogeneous systems (e.g., swarm robotic algorithms [4, 49, 42]) implement the actions used for task execution. Actions of individuals do not have to be planned in such algorithms, but only to be triggered in the form of local rules that collectively produce a desired emergent effect. Our objective is to create plans for ScORe missions on ensemble instead of agent level. These tasks then should be executed collectively by using SA-SO

abilities of the ensemble, where possible (otherwise, we have to fall back on classic planning). This strategy facilitates the act of planing for collectives so that plans are calculated straightforward and adapted at run time [28].

Summed up, our multi-robot programming framework for multipotent systems eases the act of programming, employing, and maintaining ensembles.

10.3 Methodologies

To implement our approach that comes by the drawbacks introduced in Section 10.2, we propose a layered software architecture to be implemented by all robots in the ensemble. This architecture supports all necessary functionalities needed to enable a multipotent system. Each layer encapsulates its own concepts and algorithms designed to enable SA-SO (cf. Fig 10.2). In the following, we first introduce our architecture including a specification of purposes for each of its layer. Second, we give a brief overview of algorithm classes needed to enable SA-SO. For all algorithms, we assume that robots are able to communicate without restrictions (i.e., through wireless connections such as WiFi or LTE).

10.3.1 Layered Multi-Agent System Architecture

To enable modularity and adaptability needed to come by *D1 – D3* with SA and SO (cf. Section 10.2), we model each of our robots as a *Jadex Active Components Framework Platform* [5] and denote it as *agent* $a \in A$ of our ensemble A. Besides offering a communication middleware convenient for many SA and SO functionalities, this framework offers the possibility to dynamically load and unload components from a platform. We exploit this feature to build the core feature of our approach: Dynamically exchanging capabilities of robots. From a software perspective, each of the robots in the ensemble is equal to the others (homogeneity at design time). We show the representative software design of one of these platforms (robot a_1) in Fig. 10.2. In this layered architecture, we assign special functionalities to each layer, dedicated to solving the user-defined task on a different level of abstraction. In Fig. 10.2, we further depict needed communication in terms of inter-platform communication with other system participants $\{a_1, ..., a_{n+1}\}$ and communication with the system's user that introduces the current ScoRe mission. To obtain modularity, communication is restricted to only happen between vertically (intra-/ inter-platform) and horizontally adjacent layers (inter-platform). In the following the different layers, their functionality and interaction is presented in detail.

Task Layer: The *task layer* serves as relay between the system's user and the ensemble. As the user defines a new ScORe mission, this mission is broadcasted to the ensemble (no dedicated robot has to be selected by the user). Robots receiving the mission will start to analyse it and try to decompose ScORe tasks for the ensemble.

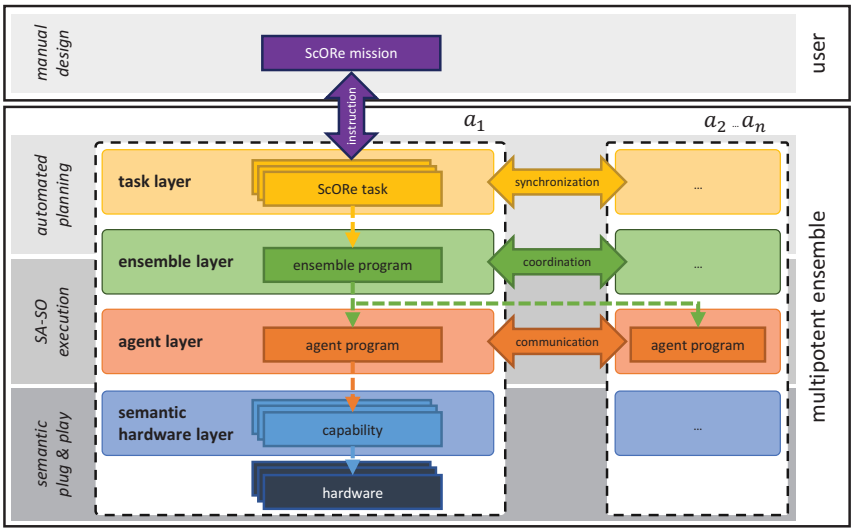


Fig. 10.2: Layered system architecture focusing on robot a_1 of a multipotent ensemble A and its interactions (solid arrows) with the user as well as other agents $\{a_2, ..., a_n\} \in A$. Dashed arrows symbolise an instantiation on the ensemble, agent, and capability layer, valid for executing a ScORe mission. For a specific ScORe mission, typically multiple ScORe Tasks have to be solved. Ensemble programmes coordinate multiple agents. Agent programmes may rely on multiple capabilities which themselves may require multiple hardware modules.

Therefore we use Hierarchical Task Networks (HTN) [15] that are especially handy for autonomous systems deployed in real-world environments. In an adapted version we propose, HTN offer a promising combination of external user control (through designed decomposition of complex tasks into primitive tasks) and an appropriate degree of freedom to enable autonomous decisions within the executing ensemble. Consequently, ScORe missions are represented as complex tasks in the HTN while the user-defined task is the root node of the network. After one robot has achieved a valid decomposition, information about the resulting ScORe tasks (i.e., the primitive tasks in the HTN) is distributed within the ensemble. This step is asynchronously done in parallel by all robots and thus multiple (different) decompositions may occur. The ensemble has to synchronise results and determine one solution to be executed (cf. Fig. 10.2). Execution then takes place in the *ensemble layer* and its subordinate layers.

Ensemble Layer: For every ScORe task there is at least one appropriate action available to solve it on ensemble layer, called *ensemble programme*. An ensemble programme encapsulates the logic for the cooperative execution of the associated ScORe task. We implement ensemble programmes as SA-SO algorithms when possible, e.g., associating swarm algorithms like Particle Swarm Optimisation (PSO) [53] to ScORe tasks of class S. If no appropriate swarm algorithm can be found, we

fall back on classical multi-agent planning [40] to create valid action sequences for solving the ScORe task. Thereby, we often exploit the advantage SA-SO offers compared to traditional multi-agent planning [28]: Actions of participants do not have to be planned but are deposited as local rules on the *agent layer* (e.g., despite every robot in the ensemble is working only according to its local rules of the PSO, the collective achieves to find a global optimum as an emergent effect of the cooperation). The ensemble layer thereby is responsible for coordinating the distributed execution. To increase robustness, the existence of multiple ensembles in parallel is possible which work on the same or on a different task simultaneously. Additionally, we allow horizontal inter-platform coordination on ensemble layer, e.g., to exchange agents among ensembles when necessary. To enable ensemble programmes to be suitable for not only one special ScORe task but for a whole class, we design them to be parameterisable. All ScORe tasks of class S, e.g., can be handled by the same algorithm where the parameter of interest can be injected. For instance, the source of gas X in gas accidents like in Fig. 10.1, the highest temperature in case of fire accidents, or highest pollen concentration in an environment monitoring scenario.

Agent Layer: The responsibilities on the agent layer are twofold. To execute ensemble programmes, agents in an ensemble have to cooperate with each other. Thus, while coordination of those cooperations is handled on the ensemble layer, the agent layer is responsible for executing the appropriate local rules. Therefore, for all ensemble programmes an accompanying *agent programme* containing those local rules is implemented on agent layer. As exchanging information between cooperating agents is often necessary within SA-SO algorithms (e.g., exchanging measurements in PSO), we allow horizontal inter-device communication on the agent layer. The second responsibility of the agent layer is to maintain a robot's self-awareness on its current capabilities and thereby its competence to participate in concretely parametrised SA-SO algorithms (PSO for gas X obviously needs participating robots to provide the *capability* to measure gas X). Thereby, the agent layer together with the ensemble layer and *capability layer* is responsible to detect situations that need the agent or even the whole ensemble to be reconfigured concerning their provided capabilities.

Capability Layer: The capability layer encapsulates the self-awareness functionalities of each robot. Enriched by a common knowledge base that stores relations between hardware modules and capabilities with robot specific knowledge (e.g., concerning a robot's geometric design), the capability layer manages the set of available capabilities in any possible robot configuration. This information is essential for the agent layer to determine its competence of participating in a coalition for a specific SA-SO algorithm. Further, this information can be used to propose beneficial reconfigurations of the robots' hardware in situations where this is needed. Moreover, the capability layer is responsible for executing capabilities, i.e., is the adapter between software representation and hardware execution. When an agent programme on the agent layer triggers the execution of a capability, the capability layer forwards this to the appropriate hardware, collects up possible callbacks (e.g., measurements), and informs the agent programme about the execution status. Thereby, the capabil-

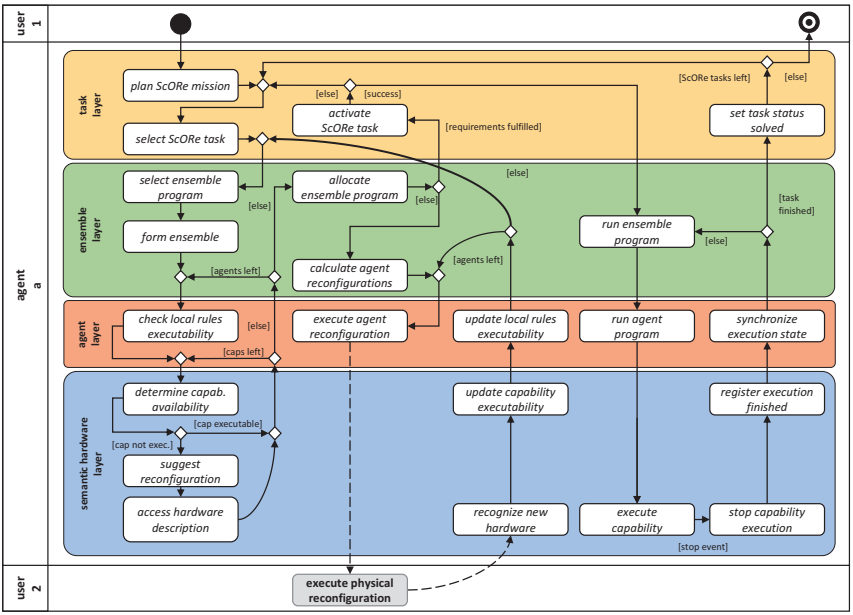


Fig. 10.3: Example of a ScORe mission execution. Activities on the task and ensemble layer are executed only on a coordinating robot while activities on the agent, the capability, and the hardware layer are executed on all devices. Currently, the physical reconfiguration has to be accomplished by a user (dashed lines) that also introduces new ScORe missions to start the overall activity.

ity layer encapsulates concrete hardware implementations from their usage in higher layers.

Semantic Hardware Layer: To enable the capability layer to deduce available capabilities from certain hardware configurations, as well as to realise actual execution on real hardware, the *semantic hardware layer* is implemented as an adapter to the hardware. The implementation of that layer is not in the scope of this work, nonetheless, the interface to it is presented in [50].

10.3.2 Algorithms and Technologies

To enable SA-SO abilities of the ensemble and increase its autonomy, we integrate algorithms from three classes that empower the agent to fulfill its responsibilities defined in the different software layers.

Basic Ensemble Algorithms: We identified algorithms for forming coalitions (i.e., ensembles) consisting of appropriately configured robots for given tasks with a market-based algorithm in previous work [30] (cf. *form ensemble* on ensemble layer

in Fig. 10.3). Moreover, we suggested to resolve conflict situations (e.g., multiple coalitions are found while only one shall activate the task, cf. *activate ScORe task* on the task layer in Fig. 10.3) by leader election in previous work [30] that can also be achieved in a distributed fashion, e.g., formulated as consensus problem [21], which we are currently investigating in. To enable the self-awareness functionalities of robots (e.g., needed for *determine capability availability* on capability layer in Fig. 10.3), we design capabilities as agents within the Jadex Active Components Framework [5] combined with semantically annotated hardware modules [50]. To let robots safely cooperate, e.g., within bound areas, we adopt the collision avoidance algorithm of [3] that works distributively and with only local sensors.

Planning and Executing ScORe missions: To enable HTN-planning on the task layer, we provide an appropriate domain description language working with actions (ensemble programmes) on the ensemble layer. This enables the design of reusable, parameterisable HTNs which can be composed for different ScORe missions in a modular fashion. To execute ensemble programmes, we similarly classify them according to the ScORe task class they are associated with. Thereby, we develop ensemble programmes implementing appropriate SA-SO algorithms for each class (cf. *select ensemble programme* on ensemble layer in Fig. 10.3). For S, we identify PSO to be fitting for continuously distributed parameters like gas or temperature. For discretely distributed parameters (e.g., objects) an adapted potential field method [27] can be used where entities distribute equally in a given area to enable the observation of the whole area. The potential field method is also a candidate to be implemented for ScORe tasks of class cO. When, e.g., a cloud of gas X has to be observed, observing robots push each other off as much as possible while additionally staying where a given gas concentration can be measured. Thereby, the tracking of the gas cloud can be seen as an emergent effect. Algorithms for the ScORe class of Re are very specific, as an infinite set of possible situations may happen during S and cO. We, therefore, identify reactions that are commonly needed in most of the situations and find appropriate SA-SO algorithms to implement them, e.g., formation flight based on local rules for guided boiding [41] (control-one-move-all) to evacuate all robots from an area.

Robust and Autonomous System: For achieving robustness and autonomy in the ensemble, the ensemble has to be able to compensate for failures of robots as well as failures in hardware providing capabilities to robots. For solving the first, we currently develop an approach to compensate for hard failures (break down of robots) as well as for soft failures (battery depletion) with a transactional task execution system using an extended coalition formation algorithm that allows for exchanging robots among ensembles. For providing the second, we developed a portfolio approach (cf. *resource allocation portfolio approach* in Fig. 10.4) for solving the multi-agent resource allocation problem (MARA) [7] to reconfigure robots within the ensemble (cf. *calculate agent reconfigurations* on ensemble layer in Fig. 10.3) that combined with a task allocation algorithm (cf. *task allocation* and *task assignment* in Fig. 10.4) maintains the ensembles operability (cf. Fig. 10.4 [25]). Thereby, we are able to determine resource (hardware modules) allocations in the ensemble to fulfill all task

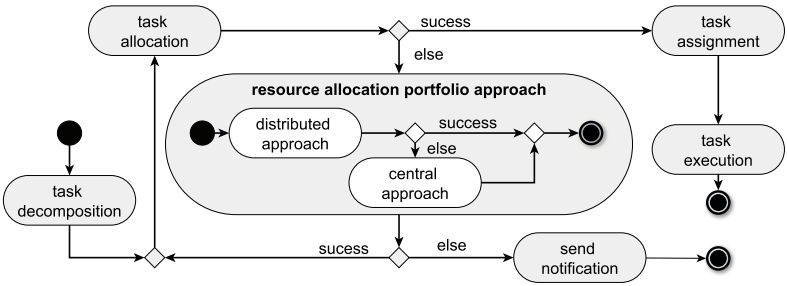


Fig. 10.4: An integrated approach for task and resource allocation (from [25]). Resource allocation takes place when task allocation fails to re-establish a productive system configuration (allocate, assign, and execute tasks). We use a portfolio approach to select an appropriate mechanism (centralised or distributed).

requirements concerning needed capabilities that have to be provided by robots (we exploit the possibility to decompose the calculation of a solution to the resource allocation problem like previously done in the energy domain [29]). Consequently, when there is no valid solution to the task allocation problem, the ensemble is reconfigured to allow a subsequently restarted task allocation (cf. *task allocation* in Fig. 10.4), assign (cf. task assignment in Fig. 10.4) and execute (cf. *task execution* in Fig. 10.4) the tasks afterward (cf. the situations marked with R in Fig. 10.1, where the systems needs to be reconfigured according to changed task requirements).

10.4 Evaluation

We already achieved some proof of concepts for the practicability of our approach in simulation as well as in real-world experiments.

Real World Experiments: In the domain of environmental research, we evaluated our system in its respective state during ScaleX Campaigns 2015 [51] and 2016 [23, 26] and demonstrated its functionalities during the FAS*2016 conference [30, 24]. During ScaleX 2016, an ensemble of up to five mobile robots were autonomously coordinated by our software to achieve a large-scale temperature profile with the Distributed Temperature Sensing (DTS) technology. Thereby, we proofed our concept of controlling robots with Jadex as well as a set of basic ensemble algorithms (self-awareness, coalition formation, leader election, and cooperative execution [30]). For realising the experiment, we instructed the ensemble to cooperatively carry a fiberglass cable along which temperature measurements can be made within every 10 centimeters. The cable was carried along a pre-scripted route in the experiment area in a height of approximately 50 meters autonomously by a heterogeneous ensemble consisting of quadrocopters as well as mobile ground robots. Task allocation, assignment, and the coordinated execution were performed in a SA-SO fashion,

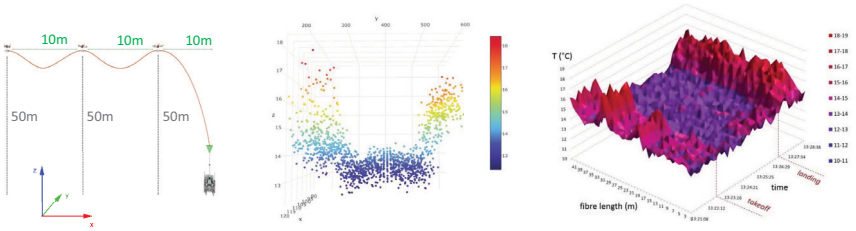


Fig. 10.5: Proof of concept from the environmental measurement campaign, where a heterogeneous ensemble consisting of four mobile robots achieved an airborne large-scale temperature measurement using a fiberglass cable for Distributed Temperature Sensing measurements in a height of approximately 50 meters.

respecting the robots different available capabilities (quadrocopters are able to move to positions higher than 1 meter, mobile ground robots are not) and the tasks required capabilities (some tasks needed to reach up to 50 meters). A sketch of the experiment setup including a proof of concepts achieved in this experiment can be found in Fig. 10.5.

Simulated Experiments: We also achieved some first results in simulation concerning the use of suitably applied and parameterisable swarm algorithms for the ScORe task pattern. Figure 10.6 shows the progress of executing the particle swarm optimisation algorithm in our simulation environment that can be applied to S tasks in the course of a ScORe mission. The robots, therefore, need to be able to freely move in 3-dimensional space ("flying") as well as measuring the parameter of interest. Robots use their self-awareness ability on the capability layer (cf. Section 10.3.1) to determine their qualification for the task. Measurements are propagated in the ensemble and the moving vector is adjusted according to the PSO rules (information on the currently measured maximum concentration of the parameter, the local concentration at a robot's current position, and a certain random component to avoid local optima). Thereby, robots that cooperate on the agent layer by executing the appropriate PSO-specific agent programme collectively execute the PSO-specific ensemble programme that is coordinated on the ensemble layer. This leads to the correct execution of a S task on task layer, requiring. In addition, we also achieved first results in executing a potential field algorithm suitable for continuous observations and guided boiding suitable for react tasks (cf. Section 10.3.2) in our simulation environment.

10.5 Related Work

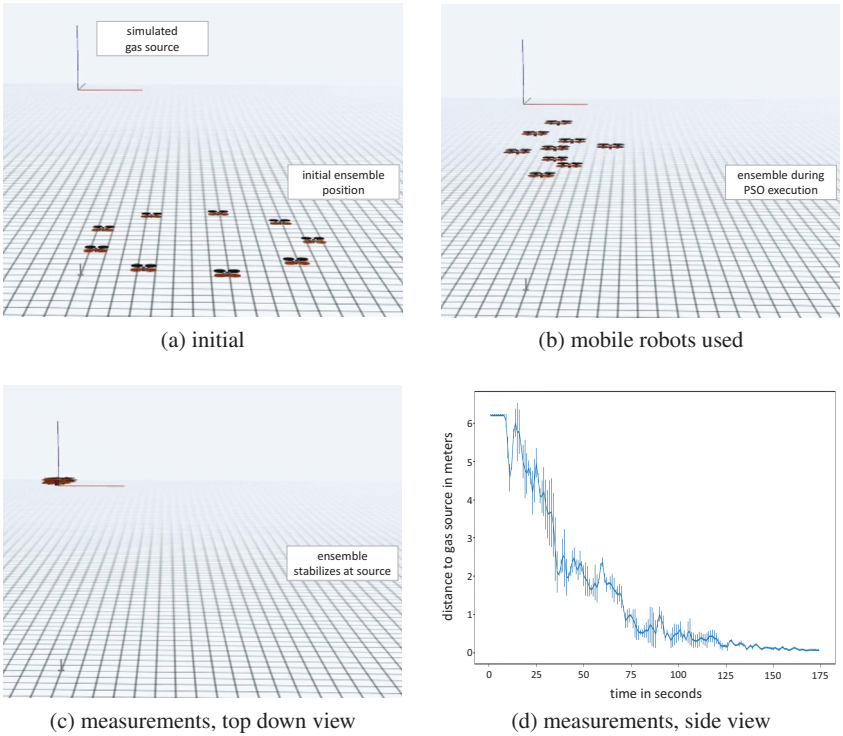
Many projects investigate mobile agents and how they can support emergency management (e.g., *Search and Rescue (SAR)* scenarios) or environmental research, which

both can be classified as ScORe missions. In the following, we discuss related approaches and point out their potential as well as their draw-backs in the context of applying them for handling ScORe missions.

Current approaches can roughly be classified into two groups: **(1)** approaches, that try to map principles found in nature to the development of systems consisting of a multitude of simple robot agents (called *Swarm Robotics* [4]), and **(2)** approaches, that use traditional AI techniques (e.g., planning, reasoning, learning, scheduling, etc.) [43, 2, 22]. Approaches of group **(1)** are characterised by the fact that agents only have very limited capabilities and are highly specialised for certain tasks which limits their applicability for ScORe missions, as different ScORe tasks typically require agents with heterogeneous capabilities.

Projects following **(1)** can mainly be found in the domain of environmental/climate research. Here, the usage of swarm robotic algorithms is more common than in SAR scenarios as these systems are often only used to collect data but not to make complex online decisions based on it. For instance, in CoCoRo (2011 - 2014) [44] as well as in its subsequent project subCULTrob (since 2015) [46] autonomous swarms of *unmanned underwater vehicles (UUVs)* and *unmanned surface vehicles (USVs)* (all specialised in their capabilities) use swarm robotic principles for ecological monitoring, search, maintenance, exploration, and harvesting in an underwater environment. Other projects focus on developing mechanisms for solving single specialised tasks, e.g., distributed environmental monitoring using mobile USVs. In this domain, the NAMOS project (2003 - 2007) [6] uses coordination done by a multi-agent system, whereas the project CORATAM (since 2014) [12] relies on controllers synthesised with evolutionary algorithms. All approaches classified in type **(1)** have in common that they reach their goals by using algorithms relying on very simple rules (e.g., finding consensus [48], cooperative transport [11, 36], or clustering [12]), that can only be achieved by the collective as a whole but not by single agents (often described as *emergent effect* [19]). On the one hand, this emergence brings robustness and fault tolerance, as misbehaviour of single agents can easily be compensated for within a homogeneous swarm where all agents are identical. On the other hand, in complex applications involving multiple tasks (e.g., ScORe missions), problems occur after achieving the goal the swarm robotic system is trained for, as these systems rely on additional guidance if interdependent decisions or actions must be made [4]. In addition, agents used in subsequent tasks usually need to have different capabilities which is often not considered in these approaches, as the swarms usually consist of homogeneous agents.

Approaches of group **(2)** appear to be more appropriate for complex scenarios. The Swarmanoid project (2006 - 2010) [10] combines multiple mechanisms from **(1)**, e.g., to solve a precisely pre-defined SAR mission using heterogeneous agents, but therefore relies on explicitly scripted behaviour. Other projects for handling SAR scenarios often combine intelligent autonomous agents with different capabilities for supporting humans in the SAR process. For example, the SHERPA project (2013 - 2017) [33] focuses on human-robot-interaction with UAV and unmanned ground



(a) initial

(b) mobile robots used

(c) measurements, top down view

(d) measurements, side view

Fig. 10.6: (A) shows the system previous to execution with the gas source indicated as coordinate system (top left) and all quadcopters in their initial circle formation. In (B) all quadcopters start to execute the local rules of PSO to cooperatively approach the gas source. (C) indicates the state where the gas source is found by the ensemble (no collision avoidance is respected in this simulation run). (D) shows the average distance of the quadcopters to the gas source in meters (during one execution of the PSO) including the standard deviation as error bars, continuously reduced with passed time.

vehicles (UGVs). The SWARMIX project (2011 - 2014) [14] combines human capabilities with those of animals (e.g., dogs), as well as artificial agents (e.g., UAV) to create heterogeneous SAR systems. Systems designed for SAR scenarios consisting of artificial agents only are often restricted in their autonomy, as they are only able to react to predefined events. In SINUS (2013 - 2015) [43] or AirShield (2008 - 2011) [8] a wireless connection can be autonomously established over a long distance to stream data from an area of interest to a distant operator for further evaluation. This evaluation and an appropriate reaction have to be determined by the human user of the system.

All of the described approaches have in common that they are heavily specialised for their dedicated application and lack the ability to autonomously adapt to changing requirements (i.e., new capabilities are demanded from agents), which is very important in ScORe missions. While certain approaches use SA and SO to adapt to a changing environment, the possibility of reallocating the agents' capabilities is neglected. This hinders ongoing approaches to actually adapt to changing ScORe tasks in ScORe missions or switch from one ScORe mission to another (e.g., from environmental monitoring to SAR). Our approach instead allows for both kinds of adaptation. By combining SA and SO principles with planning, we suggest a new paradigm of instructing an ensemble. Being able to reallocate capabilities seems to be the next step to enable real-world applications. We, therefore, will make this additional degree of freedom utilisable and consequently will consider it throughout all software layers. The resulting system architecture will allow robot ensembles to deal with ScORe missions in general.

10.6 Conclusion & Outlook

We presented a layered software architecture integrated with necessary algorithms to enable multipotent systems merging the benefits of both, homogeneous and heterogeneous systems. Therewith, robots implementing the proposed software, homogeneous at design time, can become heterogeneous specialists during runtime. We already realised a basic version of that architecture [32, 28] and integrated necessary algorithms for coalition formation [32], distributed leader election, information distribution [30], coordination of collective behaviours [30], self-adapting and self-organising reconfiguration of robot capabilities [25], and controlling mobile robots (UAVs and ground robots) in general [51, 30]. Our current and next steps include the design and implementation of 1) a domain specific language for specifying modular hierarchical task networks (HTN) in the domain of ScORe missions and 2) of an adaptive, distributed, and transactional algorithm to transfer task execution knowledge in the case of failure for increasing the robustness of the system. To 3) achieve consensus within the ensemble, e.g., during synchronisation of different planning results on the task layer (cf. Section 10.3.1), we will appropriately adapt a leader election approach [17] meeting all needed requirements defined of our system class. We further will automate the act of planning by enabling our system to learn HTNs [39] from data provided by experts so that learned valid plans are suggested to the systems user.

References

1. Babaoglu, O., Shrobe, H., eds.: First IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2007) 9-11 July, Boston, MA, USA. *Available at http://projects.csail.mit.edu/saso2007/, accessed on 2018-11-15* (Jul 2007)

2. Becker, M., Blatt, F., Szczerbicka, H.: A Multi-agent Flooding Algorithm for Search and Rescue Operations in Unknown Terrain, pp. 19–28. Springer Berlin (2013)
3. van den Berg, J., Lin, M., Manocha, D.: Reciprocal Velocity Obstacles for real-time multi-agent navigation. In: 2008 IEEE Int. Conf. on Robotics and Automation. pp. 1928–1935 (May 2008)
4. Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M.: Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence 7(1), 1–41 (2013)
5. Braubach, L., Pokahr, A.: Developing distributed systems with active components and Jadex. Scalable Computing: Practice and Experience 13(2), 100–120 (2012)
6. Caron, D., Stauffer, B., Darjany, L., Oberg, C., Pereira, A., Das, J., Heidarsson, H., Smith, R., Smith, E., Seubert, E., et al.: Networked aquatic microbial observing systems: An overview. Center for Embedded Network Sensing (2009)
7. Chevaleyre, Y., Dunne, P.E., Endriss, U., Lang, J., Lemaitre, M., et al.: Issues in multiagent resource allocation. Informatica 30(1) (2006)
8. Daniel, K., Dusza, B., Lewandowski, A., Wietfelds, C.: AirShield: A system-of-systems MUAV remote sensing architecture for disaster response. In: Proc. 3rd Annual IEEE Systems Conf. (SysCon) (2009)
9. De Cubber, G., Serrano, D., Berns, K., Chintamani, K., Sabino, R., Ourevitch, S., et al.: Search and rescue robots developed by the european icarus project. In: 7th Int. Workshop on Robotics for Risky Environments. Citeseer (2013)
10. Dorigo, M., Floreano, D., Gambardella, L.M., Mondada, F., Nolfi, S., Baaboura, T., et al.: Swarmanoid: A Novel Concept for the Study of Heterogeneous Robotic Swarms. IEEE RAM 20(4), 60–71 (2013)
11. Dorigo, M., Tuci, E., Groß, R., Trianni, V., Labella, T.H., Nouyan, S., et al.: The swarmbots project. In: Int. Workshop on Swarm Robotics. pp. 31–44. Springer (2004)
12. Duarte, M., Costa, V., Gomes, J., Rodrigues, T., Silva, F., Oliveira, S.M., Christensen, A.L.: Evolution of Collective Behaviors for a Real Swarm of Aquatic Surface Robots. PLoS ONE 11(3), 1–25 (03 2016)
13. Erol, K., Hendler, J., Nau, D.S.: HTN planning: Complexity and expressivity. In: AAAI. vol. 94, pp. 1123–1128 (1994)
14. Flushing, E.F., Gambardella, L.M., Caro, G.A.D.: A mathematical programming approach to collaborative missions with heterogeneous teams. In: 2014 IEEE/RSJ Int. Conf. on Intell. Robots and Systems
15. Georgievski, I., Aiello, M.: An Overview of Hierarchical Task Network Planning. CoRR abs/1403.7426 (2014), http://arxiv.org/abs/1403.7426
16. Gerkey, B.P., Matarić, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. The Int. Journ. of Robotics Res. 23(9), 939–954 (2004)
17. Gharehchopogh, F.S., Arjang, H.: A survey and taxonomy of leader election algorithms in distributed systems. Indian Journ. of Science and Technology 7(6), 815 (2014)
18. Giorgetti, A., Marchetto, M.C., Li, M., Yu, D., Fazzina, R., Mu, Y., et al.: Cord blood-derived neuronal cells by ectopic expression of Sox2 and c-Myc. Proceedings of the National Academy of Sciences 109(31), 12556–12561 (2012)
19. Goldstein, J.: Emergence as a construct: History and issues. Emergence 1(1), 49–72 (1999), http://dx.doi.org/10.1207/s15327000em0101_4
20. Güdemann, M., Nafz, F., Ortmeier, F., Seebach, H., Reif, W.: A specification and construction paradigm for organic computing systems. In: 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems. pp. 233–242 (Oct 2008)
21. Hamann, H.: Towards swarm calculus: urn models of collective decisions and universal properties of swarm performance. Swarm Intelligence 7(2), 145–172 (2013)

22. Hussein, A., Adel, M., Bakr, M., Shehata, O.M., Khamis, A.: Multi-robot task allocation for search and rescue missions. Journ. of Physics: Conf. Series 570(5), 052006 (2014), http://stacks.iop.org/1742-6596/570/i=5/a=052006
23. ISSELabs: Flying robot ensemble in action at the ScaleX 2016 geographic measurement campaign. *Available at https://youtu.be/MWNyUymtNSs, accessed on 2018-11-15* (Oct 2018), https://youtu.be/MWNyUymtNSs
24. ISSELabs: Saso 2016 - decentralized coordination of heterogeneous ensembles using jadex. *Available at https://youtu.be/G8JHShUlQY0, accessed on 2018-11-15* (Oct 2018), https://youtu.be/G8JHShUlQY0
25. J.Hanke, Kosak, O., Schiendorfer, A., Reif, W.: Self-organized Resource Allocation for Reconfigurable Robot Ensembles. In: 2018 IEEE 12th Int. Conf. on Self-Adaptive and Self-Organizing Systems (Sept 2018)
26. KIT IMK/IFU, G.P.: Scalex. *Available at https://scalex.imk-ifu.kit.edu/, accessed on 2018-11-15* (Oct 2018), https://scalex.imk-ifu.kit.edu/
27. Koren, Y., Borenstein, J.: Potential field methods and their inherent limitations for mobile robot navigation. In: Robotics and Automation, 1991. Proceedings., 1991 IEEE Int. Conf. on. pp. 1398–1404. IEEE (1991)
28. Kosak, O.: Facilitating Planning by Using Self-Organization. In: 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W). pp. 371–374 (Sept 2017)
29. Kosak, O., Anders, G., Siefert, F., Reif, W.: An Approach to Robust Resource Allocation in Large-Scale Systems of Systems. In: Self-Adaptive and Self-Organizing Systems (SASO), 2015 IEEE 9th Int. Conf. on. pp. 1–10 (2015)
30. Kosak, O., Wanninger, C., Angerer, A., Hoffmann, A., Schierl, A., Seebach, H.: Decentralized Coordination of Heterogeneous Ensembles Using Jadex. In: 2016 IEEE 1st Int. Workshops on Found. and Applications of Self* Systems (FAS*W). pp. 271–272 (2016)
31. Kosak, O.: A Decentralised Swarm Approach for Mobile Robot-Systems. In: Organic Computing: Doctoral Dissertation Colloquium 2015. vol. 7, p. 53. kassel university press GmbH (2015)
32. Kosak, O., Wanninger, C., Angerer, A., Hoffmann, A., Schiendorfer, A., Seebach, H.: Towards Self-Organizing Swarms of Reconfigurable Self-Aware Robots. In: Found. and Appl. of Self* Sys., IEEE Int. Workshops on. pp. 204–209. IEEE (2016)
33. Marconi, L., Leutenegger, S., Lynen, S., Burri, M., Naldi, R., Melchiorri, C.: Ground and aerial robots as an aid to alpine search and rescue: Initial sherpa outcomes. In: Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE Int. symposium on. pp. 1–2. IEEE (2013)
34. Martinez, C., Sampedro, C., Chauhan, A., Campoy, P.: Towards autonomous detection and tracking of electric towers for aerial power line inspection. In: 2014 Intern. Conf. on Unmanned Aircraft Systems (ICUAS). pp. 284–295 (2014)
35. Matikainen, L., Lehtomäki, M., Ahokas, E., Hyyppä, J., Karjalainen, M., Jaakkola, A., et al.: Remote sensing methods for power line corridor surveys. ISPRS Journ. of Photogrammetry and Remote Sensing 119(Supplement C), 10 – 31 (2016)
36. Mondada, F., Gambardella, L.M., Floreano, D., Nolfi, S., Deneuborg, J.L., Dorigo, M.: The cooperation of swarm-bots: physical interactions in collective robotics. IEEE Robotics Automation Magazine 12(2), 21–28 (2005)
37. Müller-Schloer, C., Tomforde, S.: Organic Computing – Techncial Systems for Survival in the Real World. Autonomic Systems, Birkhäuser Verlag (October 2017), iSBN: 978-3-319-68476-5

38. Murphy, R.R., Tadokoro, S., Nardi, D., Jacoff, A., Fiorini, P., Choset, H., Erkmen, A.M.: Search and Rescue Robotics. In: Siciliano, B., Khatib, O. (eds.) Handbook of Robotics. Springer (2008)
39. Nejati, N., Langley, P., Konik, T.: Learning hierarchical task networks by observation. In: Proceedings of the 23rd Int. Conf. on Machine Learning. pp. 665–672. ICML '06, ACM, New York, NY, USA (2006)
40. Nissim, R., Brafman, R.I., Domshlak, C.: A general, fully distributed multi-agent planning algorithm. In: Proc. of the 9th Intern. Conf. on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1. pp. 1323–1330. AAMAS '10, Intern. Foundation for Autonomous Agents and Multiagent Systems (2010), http://dl.acm.org/citation.cfm?id=1838206.1838379
41. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. ACM SIGGRAPH computer graphics 21(4), 25–34 (1987)
42. Rubenstein, M., Cornejo, A., Nagpal, R.: Programmable self-assembly in a thousand-robot swarm. Science 345(6198), 795–799 (2014)
43. Scherer, J., Yahyanejad, S., Hayat, S., Yanmaz, E., Andre, T., Khan, A., et al.: An Autonomous Multi-UAV System for Search and Rescue. In: Proc. of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use. pp. 33–38. DroNet '15, ACM, Florence, Italy (2015)
44. Schmickl, T., Thenius, R., Moslinger, C., Timmis, J., Tyrrell, A., Read, M., et al.: Cocoro – the self-aware underwater swarm. In: 2011 Fifth IEEE Conf. on Self-Adaptive and Self-Organizing Systems Workshops. pp. 120–126 (2011)
45. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Art. Intelligence 101(1), 165 – 200 (1998), http://www.sciencedirect.com/science/article/pii/S0004370298000459
46. Thenius, R., Moser, D., Kernbach, S., Kuksin, I., Kernbach, O., Elena Kuksina, E., et al.: subCLUTron: a learning, self-regulating, self-sustaining underwater society/culture of robots. Art. Life and Intell. Agents Symposium, 2016 (2016)
47. Tomforde, S., Sick, B., Müller-Schloer, C.: Organic Computing in the Spotlight. arXiv.org (January 2017), http://arxiv.org/abs/1701.08125
48. Valentini, G., Hamann, H., Dorigo, M.: Efficient decision-making in a self-organizing robot swarm: On the speed versus accuracy trade-off. In: Proc. of the 2015 Int. Conf. on Autonomous Agents and Multiagent Systems. pp. 1305–1314. AAMAS '15, Int. Found. for Autonomous Agents and Multiagent Systems, Istanbul, Turkey (2015), http://dl.acm.org/citation.cfm?id=2772879.2773319
49. Vásárhelyi, G., Virágh, C., Somorjai, G., Tarcai, N., Szörényi, T., Nepusz, T., Vicsek, T.: Outdoor flocking and formation flight with autonomous aerial robots. In: 2014 IEEE/RSJ Int. Conf. on Intell. Robots and Systems. pp. 3866–3873 (2014)
50. Wanninger, C., Eymller, C., Hoffmann, A., Kosak, O., Reif, W.: Synthesising Capabilities for Collective Adaptive Systems from Self-Descriptive Hardware Devices - Bridging the Reality Gap. In: 8th Int. Symp. On Leveraging Appl. of Formal Methods, Verification and Validation (Sept 2018)
51. Wolf, B., Chwala, C., Fersch, B., Garvelmann, J., Junkermann, W., Zeeman, M.J., et al.: The SCALEX Campaign: Scale-Crossing Land Surface and Boundary Layer Processes in the TERENO-preAlpine Observatory. Bulletin of the American Meteor. Soc. 98(6), 1217–1234 (2017)
52. Zhang, J., Liu, L., Wang, B., Chen, X., Wang, Q., Zheng, T.: High Speed Automatic Power Line Detection and Tracking for a UAV-Based Inspection. In: 2012 Intern. Conf. on Industrial Control and Electronics Engineering. pp. 266–269 (2012)

53. Zhang, Y., Wang, S., Ji, G.: A comprehensive survey on particle swarm optimization algorithm and its applications. Mathematical Problems in Engineering 2015 (2015)