

Engineering Swarms of Cyber-Physical Systems

Melanie Schranz,
Wilfried Elmenreich, and
Farshad Arvin



CRC Press
Taylor & Francis Group

A SCIENCE PUBLISHERS BOOK

Engineering Swarms of Cyber-Physical Systems

Melanie Schranz

Lakeside Labs, Klagenfurt, Austria

Wilfried Elmenreich

University of Klagenfurt, Klagenfurt, Austria

Farshad Arvin

Durham University, Durham, UK



CRC Press

Taylor & Francis Group

Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

A SCIENCE PUBLISHERS BOOK

First edition published 2026
by CRC Press
2385 NW Executive Center Drive, Suite 320, Boca Raton FL 33431

and by CRC Press
4 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

© 2026 Taylor & Francis Group, LLC

CRC Press is an imprint of Taylor & Francis Group, LLC

Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, access www.copyright.com or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. For works that are not available on CCC please contact mpkbookspermissions@tandf.co.uk

Trademark notice: Product or corporate names may be trademarks or registered trademarks and are used only for identification and explanation without intent to infringe.

Library of Congress Cataloging-in-Publication Data (applied for)

ISBN: 978-1-032-04715-7 (hbk)

ISBN: 978-1-032-04733-1 (pbk)

ISBN: 978-1-003-19446-0 (ebk)

DOI: 10.1201/9781003194460

Typeset in Times New Roman
by Prime Publishing Services

Preface

In the evolving landscape of modern technology, Cyber-Physical Systems (CPS) represent the forefront of innovation, blending the physical and digital worlds seamlessly. However, as these systems become increasingly interconnected and complex, new challenges arise. In complex systems composed of numerous interacting components, the necessity for adaptation or expansion often becomes evident. Minor adjustments, such as a simple software update, might subtly alter component behavior, undermining key assumptions about the broader system. How can we design systems that are not only powerful but also robust, adaptable, and scalable? Conventional top-down design approaches often fall short meeting these requirements, particularly when addressing the ongoing design, operation, and maintenance of such complex systems. One promising answer lies in the fascinating field of swarm intelligence.

Swarm systems utilize simple local interactions to produce sophisticated, emergent behaviors similar to those observed in natural entities like flocks of birds or colonies of ants. Inspired by this potential, *Engineering Swarms for Cyber-Physical Systems* aims to guide readers through the entire journey of applying swarm intelligence in CPS, from conceptual modeling to final deployment. This book is structured to be a one-stop resource for engineers, researchers, and students. It begins with an introduction to swarm intelligence, illustrating its vast potential through examples from robotics, manufacturing, and search-and-rescue operations. We then explore the core principles of modeling, design methods—including machine-learning approaches—and simulation techniques. A highlight of the book is its hands-on character: programming examples and practical insights are woven throughout, enabling you to translate theoretical concepts into real-world applications effortlessly.

Our aspiration is that this book will serve not only as a technical guide but also as a source of inspiration. Whether you are an experienced engineer or a curious researcher, you will find the tools and knowledge needed to create innovative swarm-based CPS solutions.

Thank you for taking this journey with us. Together, let's explore the future of engineering swarms and unlock the potential of intelligent, adaptive systems. Happy designing!



Taylor & Francis

Taylor & Francis Group

<http://taylorandfrancis.com>

Contents

<i>Preface</i>	iii
<i>List of Figures</i>	viii
<i>List of Tables</i>	xii
<i>List of Acronyms</i>	xiii
1. Swarm Intelligence in Cyber-Physical Systems	1
1.1 Basic Terminology	1
1.1.1 Swarm-Intelligent Behavior	3
1.1.2 A Taxonomy on Biologically-Inspired Swarm Intelligence Algorithms	4
1.1.3 Multi-Agent vs. Swarm Systems	6
1.2 Motivation to Engineer Swarm Intelligence for CPSs	7
2. Principles of Swarm Intelligence	10
2.1 Micro and Macro Level	10
2.2 Emergence	11
2.3 Feedback	12
2.4 Scalability	12
2.5 Adaptivity	13
2.6 Robustness	13
2.7 Communication	14
2.8 Superlinearity	15
2.9 Exploration and Exploitation	18

3. Modeling Swarms of Cyber-Physical Systems	20
3.1 Modeling Principles	21
3.1.1 Challenges	22
3.2 Use Case: Models in the Production Plant	24
3.2.1 Swarm Member Candidates	25
3.2.2 Network Model	26
3.3 Use Case: Models in Search and Rescue Applications	26
3.3.1 Model-Driven Engineering of CPS Swarms	27
3.3.2 Hardware Modeling	28
3.3.3 Behavior Modeling	28
3.3.4 Code Generation	32
3.3.5 Modeling on the Example of the SAR Use Case	32
3.4 Use Case: Models in Edge Computing	38
3.4.1 Modeling Agents in the Edge Continuum	40
3.4.2 Challenges in Modeling Agents for EMDCs	41
4. Engineering Swarm Behavior	43
4.1 Basic CPS Swarm Behavior	43
4.2 Use Case: Collective Motion	46
4.3 Use Case: Strategies to Mislead UAV Swarms	50
4.4 Use Case: The Principle of Hormones for Production Plants	52
4.5 Design Behaviors using the Concept of Evolution	56
4.6 Framework for Evolutionary Design	58
4.6.1 Architecture	58
4.6.2 Graphical User Interface	62
4.6.3 Workflow	63
4.6.4 External Simulators	64
4.7 Why Evolutionary Optimization Needs Simulation	65
5. Simulating Swarms of Cyber-Physical Systems	67
5.1 Simulation Requirements	68
5.2 Abstract Simulation	70
5.2.1 SwarmFabSim: A NetLogo Implementation	71
5.3 Physics-based Simulation	75
5.3.1 BeeGround: A Simulation Platform	81
6. Swarm Robotic Platforms	84
6.1 Sensors	84
6.2 Actuators	85
6.3 Communication	87
6.3.1 Pheromone-based Communication	87

6.4	Swarm Robotic Research Platforms	93
6.4.1	Terrestrial	94
6.4.2	Aerial	98
6.4.3	Aquatic	99
6.4.4	Outer Space	100
6.5	Project: How to Build your own Swarm Robot	100
6.5.1	A Robot with Legs: Spiderino	100
6.5.2	A Wheeled Robot: Mona	103
6.5.3	Another Wheeled Robot: Mechalino	105
7.	Open Challenges and Outlook	108
7.1	Challenges in Swarms of CPSs	109
7.1.1	Challenges Deriving CPS and Swarm Properties	109
7.1.2	Challenges Designing the Local Rules	110
7.1.3	Real-World Deployment Challenges	112
7.1.4	How to Address the Challenges	113
7.2	Future Trends and Directions	113
7.2.1	Future Inspirations	114
7.2.2	Promising Future Applications	117
7.2.3	Research Challenges	126
	<i>Bibliography</i>	129
	<i>Index</i>	169

List of Figures

1.1	Taxonomy of nature-inspired swarm intelligence algorithms.	5
2.1	The advantages of a swarm: Adaptivity, robustness, and scalability (adapted from [292]).	11
2.2	Example of an artificial pheromone system, COSΦ [18], that was developed for swarm robotic applications. In this scenario, a leader is releasing pheromones and five followers are following the leader. The pheromone trail is evaporating over time.	15
2.3	The swarm performance illustrated as a function of the number of swarm members, extracted from various experiments [37, 153]. Line (a) shows the hypothetically linear scale of a disembodied algorithm with swarm size. Line (b) presents the logarithmic scaling of a disembodied algorithm that requires shared resources among its instances when implemented for a parallel execution. Finally, line (c) shows the typical scaling of physically embodied swarms (or of physics-considering simulation of such agents) [333].	17
3.1	The modeling and abstraction of a self-organizing system adapted from [153].	21
3.2	The behavior library structure [345].	30
3.3	The hardware model of the UAV prototype [345].	34
3.4	The UAV SAR mission processes [345].	35
3.5	The complex SAR behavior of the UAVs [345].	36
3.6	Schematic architecture demonstrating the inter-edge resource allocation in clusters: Nodes, pool of resources and the overall edge-cloud interaction [334].	39
4.1	Taxonomy of swarm behaviors (extended from [337]).	44
4.2	The schematic description of elastic interaction (from [26]).	49
4.3	Swarm collective exploration controlled by (a) AES, (b) AESTCACS, (c) AES-GA, and (d) OCM.	50
4.4	Main idea of the scenario, in which (a) UAVs are attacking a target and (b) defenders are induced into the attacking swarm to mislead it [354].	51

4.5	GWO simulation results: (a) Probability h_p that attackers win over an increasing number of defenders and (b) probability density of achieved hits over time and increasing number of defenders [354].	53
4.6	Proposed design methodology [112].	57
4.7	Screenshot of the FREVO GUI showing the evolution of an example problem.	63
4.8	Screenshot of the source code skeleton of a newly created problem.	64
5.1	Example of a herding scenario in an abstract simulation. The positions of the dogs are shown with blue diamonds and the sheep with red circles. The target point is marked by a red star. At the beginning, the dogs were scattered around the target area. From 25 to 40 seconds, the sheep were guided towards the goal. After 40 seconds, one group of sheep had arrived and the dogs began to assist the other flock. At 50 seconds, the two flocks of sheep were combined into one large herd and the herding task was completed.	71
5.2	SwarmFabSim: Screenshot of the user interface.	73
5.3	SwarmFabSim architecture [390].	74
5.4	Physics-based models of some of the available robot libraries in Webots software; (left) large and (right) small mobile robots.	76
5.5	(a) Mona, an open-source, cost-effective robot designed for swarm robotics [17], and (b) a virtual model of Mona in Webots [11].	76
5.6	(a) Colias, an open-source micro-robot developed for swarm robotic applications [19], (b) a virtual model of Colias in Webots and (c) a detailed CAD model of Colias.	77
5.7	The Search & Rescue mission began at $t = 0$ s. Ten seconds later, rescuers were on their way to the virtual target. At $t = 30$ s, they had formed a pentagon around the target. By $t = 40$ s, the lost robot had been contained and was being guided home. Unfortunately, two of the rescuers (marked by White crosses) had stopped working due to an unexpected fault. At $t = 50$ s, the remaining rescuers had created a new triangular formation to guide the lost robot back to the starting point [169].	78
5.8	(a) The system architecture delineating the proposed human-swarm Interaction utilizing Omnipotent Virtual Giant, as detailed in Jang et al. [178]. Experimental validation showcasing (b) the relocation of holographic objects of robots and (c) the subsequent movement of real robots towards these relocated objects. Dashed arrows indicate the remaining trajectory to the target objects.	78

5.9	Screenshots of simulation examples of multi-vehicle simulations using (a) Unreal Engine [414] and (b) Webots [168].	79
5.10	(a) Gazebo environment for training and (b) real-world experiment using three TurtleBot3 Waffle Pi mobile robots [170].	80
5.11	BeeGround User Interface (UI) featuring arena with a gradient cue with a large size swarm. The <i>Object Hierarchy</i> provides an overview of all the elements in the environment, such as robots, walls, obstacles, and cues. The <i>Simulation Display</i> is a graphical representation of the simulation. The <i>Configuration Window</i> allows users to adjust the settings for the BeeGround simulation, including the agents and their behavioral parameters. The <i>Asset/Model Folders</i> is a library of all the assets related to the simulation. Finally, the <i>Console/Debug Log Window</i> tracks all errors or debugging information that occur during the simulation [229].	83
5.12	An example of BEECLUST aggregation in the BeeGround with 1000 robots in an environment with a single gradient cue (from [229]).	83
6.1	An example of a swarm robotic platform, equipped with six IR sensors, where Robot A receives IR from Robot B that has a 0° orientation (from [19]).	86
6.2	An example of a multi-layer pheromone system (from [231]).	86
6.3	On the left is the Colias micro-robot and on the right is the Colias bottom board, featuring pheromone sensing capability. Various modules of Colias include A) main processor, B) IR proximity sensors, C) digital camera, D) micro-motors with gearhead, E) 22 mm wheels, F) pheromone detectors (light intensity sensors), G) battery recharging unit, H) main switch, and J) In-System Programming (ISP) programming port (from [266]).	89
6.4	The first row presents sample experiments conducted without diffusion and with fast cue speed, devoid of pheromone injection. In contrast, the second row illustrates sample experiments conducted with diffusion and fast cue speed, incorporating pheromone injection, captured at time instances $t = 0$ s, $t = 100$ s, and $t = 200$ s from left to right (from [266]).	90
6.5	The experimental setup employed for the pheromone system in Na et al. [266] encompasses a PC dedicated to tracking robots and generating pheromones, a digital camera for monitoring robot positions, a horizontally positioned 42" LCD screen, an aluminum frame encircling the arena, and the presence of Colias mobile robots.	91

6.6	Spiderino robot (left) and its components.	102
6.7	(left) Mona robot main platform. (right) The architecture of the basic platform [17].	104
6.8	The Mona main-board and various sub-modules.	104
6.9	Mechalino components: Chassis, wheel, axle, mainboard, battery.	106
6.10	Assembled Mechalino with ArUco marker on top.	107

List of Tables

3.1	The swarm functions used in the SAR mission as part of the Swarm Library [345].	37
3.2	The hardware functions used in the SAR mission as part of the Abstraction Library [345].	37
3.3	The swarm behaviors used in the SAR mission as part of the Swarm Library [345].	37
3.4	Events used in the SAR mission [345].	38
4.1	The values of the control parameters for AES, AES-TCACS, AES-GA and OCM algorithms.	49
4.2	Suggested algorithm parameters.	55
5.1	List of simulation platforms commonly used for swarm robotics.	68
6.1	Classification of research platforms for swarm robotics (adapted from Schranz et al. [337]).	101

List of Acronyms

ACO	Ant Colony Optimization
AES	Active Elastic Sheet
ANN	Artificial Neural Network
API	Application Programming Interface
BDD	Block Definition Diagram
CPS	Cyber-Physical System
CPSoS	Cyber-Physical System of Systems
CPU	Central Processing Unit
EA	Evolutionary Algorithm
FREVO	Framework for Evolutionary Design
FSM	Finite State Machine
GUI	Graphical User Interface
IBD	Internal Block Diagram
IC	Integrated Circuit
ID	Identifier
IoE	Internet of Everything
IoT	Internet of Things
IMU	Inertial Measurement Unit
IR	Infrared
ISP	In-System Programming
MCU	Microcontroller Unit
NEAT	NeuroEvolution of Augmenting Topologies
OS	Operating System
PCB	Printed Circuit Board
PSO	Particle Swarm Optimization
PCT	Planned Cycle Time
RFID	Radio-Frequency Identification
ROS	Robot Operating System
RPT	Raw Process Time
SAR	Search and Rescue
SCXML	State Chart XML

SoC	System on Chip
SPP	Self-Propelled Particles
SysML	Systems Modeling Language
TCACS	Tabu Continuous Ant Colony System
TMS	Traffic Management System
UAV	Unmanned Aerial Vehicle
UI	User Interface
UGV	Unmanned Ground Vehicle
UML	Unified Modeling Language
URDF	Unified Robotic Description Format
USV	Unmanned Surface Vehicle
UUV	Unmanned Underwater Vehicle
UWB	Ultra Wideband
WIP	Work in Progress
VTL	Velocity Template Language
V2X	Vehicle to Everything

Chapter 1

Swarm Intelligence in Cyber-Physical Systems

Swarm intelligence is a concept adapted from nature to complex technical systems of various application domains including smart grids, cities, or mobility, and industry 4.0. Swarms of bees, birds, fish, and other organisms demonstrate how a self-organizing behavior can solve complex tasks without a central control entity dictating rules and goals to the individual agents within the swarm. This chapter aims to give a general overview of the typical terminologies for swarm intelligence concepts in general and motivates the engineering of swarm intelligence for CPSs.

1.1 Basic Terminology

A swarm consists of individual, simple, and homogeneous agents [86]. Swarm behavior describes social animal behaviors that exhibit a strong innate or developed inclination to collaboratively achieving a common global objective, such as foraging, nest-building, or defending against enemies. Thus, swarm intelligence is based on the coordination and control mechanisms that exist in natural swarms, which operate in dynamic and diverse environments. Despite the complexity of their goals, individual agents in a swarm typically follow simple rules and interact locally with their peers and surroundings. By doing so, they generate collective behavior that enables the swarm to solve complex tasks. This makes the swarm scalable, adaptable, and robust to changing conditions.

In nature, swarm behavior is often observed in animals that exhibit eusocial behavior, like honeybees, ants, termites, and naked mole rats [275]. Eusociality is characterized by several distinctive features, including cooperative care of offspring, where individuals within the group collaborate

to nurture offspring, extending their care even to those that are not direct descendants. Additionally, eusocial species often exhibit a clear division of labor, with distinct castes or groups specializing in specific tasks within the colony. Another hallmark of eusociality is the reproductive division of labor, where reproduction is taken care of by a subset of the individuals within the group, often a reproductive queen and a few males. This specialization in reproduction ensures a well-organized and efficient social structure. Furthermore, a key characteristic is the presence of overlapping generations in the swarm. This stands in contrast to species where parents typically perish before the offspring reach maturity. These features collectively define and distinguish eusocial behavior in various animal species. Given proper abstraction, eusocial behavior can be an inspiring model for CPS swarm systems.

A very early definition of a swarm was rather simple, referring to a group of animals executing a joint movement pattern. This is still reflected in the traditional definition that can be found in the Oxford Dictionary, where a swarm is “*A large or dense group of flying insects*”. The first formal definition was given by Farley and Clark in 1954 as “a system which changes its basic structure as a function of its experience and environment” [110]. The term *swarm intelligence* was first introduced in 1989 by Beni and Wang [33]. It was used to describe the dynamics of a group of cellular robots that could execute an intelligent collective behavior. This marks the starting point at which swarm intelligent behaviors were studied outside of natural sciences [333]. Bonabeau and Meyer [40] in 2001 effectively summarized all the necessary characteristics of a natural system to call it a swarm intelligence system: “*Social insects work without supervision. Their teamwork is largely self-organized, and coordination arises from the different interactions among individuals in the colony. Although these interactions might be primitive (one ant merely following the trail left by another, for instance), taken together, they result in efficient solutions to difficult problems (such as finding the shortest route to a food source among a myriad of possible paths). The collective behavior that emerges from a group of social insects has been dubbed swarm intelligence.*” The extension of the same concepts to swarm robotics was captured by the definitions of Şahin and Spears [320] in 2005: “*Swarm robotics is the study of how a swarm of relatively simple physically embodied agents can be constructed to collectively accomplish tasks that are beyond the capabilities of a single one*” and “*Swarm robotics emphasizes self-organization and emergence while considering the issues of scalability and robustness.*”

Animals exhibiting various evolved swarm behaviors to collaboratively achieve common goals, such as foraging, nest building or defending against enemies have long inspired swarm intelligence in robotics and advancements in CPS developments. The observed swarm intelligence models gain an understanding of the principle patterns and rules that the agents execute in a natural system. This also gives insights into the conditions, rules, and interactions that lead to a swarm behavior. Moreover, research has shown that an individual member of a swarm is typically incapable of finding an

optimal solution on its own [131]. Instead, a successful solution often arises from the collective behavior of the swarm as a whole.

From a CPS perspective, an engineered swarm comprises a collection of agents that can vary in complexity from relatively simple entities like swarm robotic research platforms with small, basic robots equipped with few sensors to highly complex systems such as autonomous cars. As a swarm, these agents reach a common global goal collaboratively using relatively straightforward behavior rules, which are implemented locally by each agent. When individuals interact with each other and/or their environment following well-designed rules, they can exhibit collective behavior capable of solving complex tasks effectively. Systems designed based on swarm intelligence typically exhibit characteristics, such as parallel and distributed processing and control, scalable performance, adaptability to dynamic variations, and resilience to losses and failures of individual components [333].

1.1.1 Swarm-Intelligent Behavior

To determine whether a system is truly swarm-intelligent, we propose considering the following two cases [333].

1. If the system's functionality remains intact even when operated by a single swarm member it does, not qualify as a swarm-intelligent system. If there is only one active agent, the system retains its full set of capabilities. In this scenario, these capabilities do not depend on any swarm behavior, even if multiple agents are concurrently in operation [333]. An illustrative case is the concept of "sweeping," also known as uniform coverage, which can be achieved by a single swarm agent or multiple agents collaborating to cover a designated area. To address a coverage problem within a fixed area using a UAV swarm, a simple approach is to distribute the UAVs across the area. Each UAV independently executes a sweeping algorithm for its assigned section of the map, without any interaction with other swarm members during the algorithm's execution. The sole drawback of employing a single UAV in this context is that it might take longer to complete the task.
2. As the swarm size increases, the overall performance ratio improves, indicating the presence of a swarm-intelligent system. If the overall performance per member decreases as the swarm size increases across the entire size range, the system faces scalability issues rather than benefiting from larger population sizes. For instance, increasing the number of cars on a street boosts the capacity to transport people and goods. However, as cars interfere with each other, the incremental gain in transport capacity diminishes once a certain density is reached. Thus, in this scenario a car's operation does not benefit from the presence of additional vehicles.

A comprehensive examination of this test, along with supporting empirical data, can be found in the work by Hamann et al. [153]. In such a swarm-intelligent system, the agents reach their global objective according to locally executed rules from which the overall behavior emerges through the inter-swarm interactions. In the case of CPS swarms, the behaviors displayed by each individual CPS are based on a local set of rules. These rules can be as simple as mapping sensor inputs to actuator outputs or as complex as evaluating and analyzing local information before taking action. These behaviors require interactions with the physical world, which encompass the environment as well as other CPSs [122]. To perform these interactions, the CPS reads and interprets sensory data, processes it, and drives the actuators accordingly. This sequence of interactions is known as the basic behavior, which is repeated either indefinitely or until a desired state is achieved [333].

1.1.2 A Taxonomy on Biologically-Inspired Swarm Intelligence Algorithms

Swarm intelligence algorithms are inspired by the field of biology. Most algorithms mimic the behavior of a natural system and apply it to an engineering problem. Thus, swarm intelligence algorithms are usually referred to as nature-inspired [155]. Apart from biology, other research fields like physics and chemistry also inspire the development of algorithms. However, in the proposed taxonomy, these fields are not considered. Although these algorithms are nature-inspired they do not demonstrate swarm intelligence behavior. This also holds for evolutionary computation, self-organizing neural networks, and cellular automata. There are other frameworks that draw inspiration from nature or natural processes but are not considered swarm-intelligent. Among them are biomimetic algorithms, which rely on mimicking biological processes and models. Examples include flower pollination algorithm [416], great salmon run [259], and dolphin echolocation [196]. Other more generically nature-inspired algorithms mimic physical or chemical laws instead of biological ones. Some examples of these types of algorithms include simulated annealing [206], spiral optimization [375], water cycle algorithm [107], and galaxy-based algorithm [349]. Other notable examples include the fireworks algorithm and its variants [376], the swarm chemistry particle algorithms [323], and various variants of the harmony search algorithm [129]. These examples exhibit the characteristic properties of swarm intelligence, but their local rule set does not have a specific biological source of inspiration. Given the vast array of inspiration sources and the occasionally indistinct boundaries between algorithms, it can be challenging to pinpoint categories or processes that can be definitely identified as recognized sources of inspiration for swarm-intelligent behavior [333].

In this section, we exclude the theoretical and mathematical descriptions of individual swarm intelligence algorithms, as these have already been extensively covered by various authors including Bonabeau et al. [39],

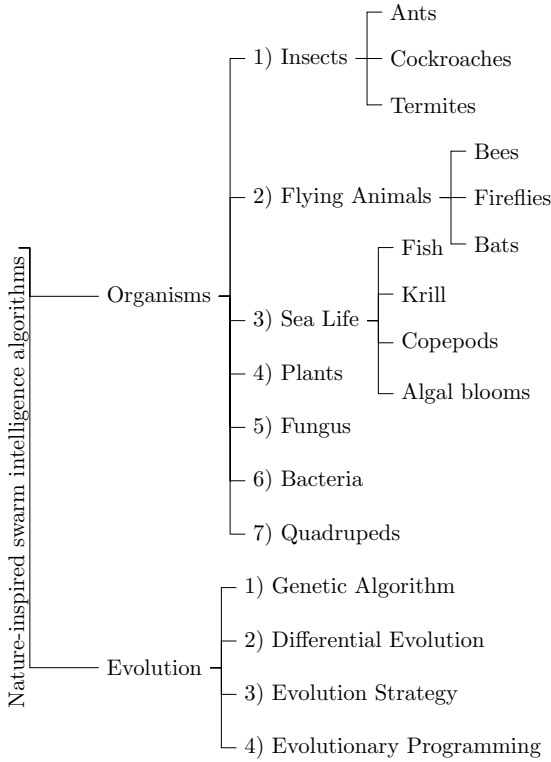


Figure 1.1: Taxonomy of nature-inspired swarm intelligence algorithms. ↵

Camazine et al. [53], Garnier et al. [131], Blum and Li [36], Floreano and Mattiussi [122], Parpinelli and Lopes [281], Binitha and Sathya [35], Yang et al. [417], Krause et al. [214], Hassanien and Emary [155], Yang et al. [418], among others.

For classifying swarm intelligence algorithms, this is not accurate enough as many nature-inspired algorithms actually do not relate to swarm behavior [145]. Therefore, we propose the taxonomy of models for existing and possible future swarm intelligence algorithms based on their sources of inspiration, as seen in Figure 1.1. The purpose of this categorization—organisms and evolution—is not to present a complete taxonomy of swarm intelligence inspirations. Instead, it aims to highlight the diversity of natural inspirations and emphasize that only a few swarm intelligence algorithms are currently suitable for use in swarms of CPSs. The biggest issue is their applicability for swarms of CPSs. For instance, Ant Colony Optimization [88] would require individual CPSs to utilize the environment for information exchange through stigmergy. Possibly future use of swarming CPSs might turn this requirement into an advantage.

Swarm intelligence is not limited to natural systems alone but can be applied to any complex system comprising multiple interacting components that can exhibit useful properties like collective decision-making, regulation, homeostasis, and periodic patterns. Thus, the key to utilizing any collective system—whether composed of living components or not—as a source of inspiration for designing a swarm-intelligent system lies in understanding why and how it functions effectively [333]. Thus, behaviors as described in Section 1.1.1 should be extracted, abstracted, and translated to the CPS swarm domain.

1.1.3 Multi-Agent vs. Swarm Systems

Both terms appear frequently in literature, making it challenging to distinguish between multi-agent and swarm systems. Characteristics typical of one system often appear in the other. A common feature is that both multi-agent systems and swarm systems consist of multiple agents or CPSs that cooperate. Additionally, attributes such as interaction types, local intelligence, and distributed knowledge are partially assigned to both systems.

Therefore, categorizing systems into a single class is often impractical. In the following sections, we will further analyze characteristics that provide sufficient evidence to classify a system as multi-agent, swarm, or, as is often the case, irrelevant in the first place.

Emergence and Homogeneity

Swarm systems are considered to be bio-inspired drawing their behavior from organisms like ants, bees and fish, etc. In such systems, we argue the swarm members are numerous, relatively simple, and homogeneous [333]. The simplicity of each agent implies that a single swarm member would be incapable of achieving the overall system’s goal independently. Additionally, the global goal is typically unknown to the individual agents. This is where the concept of emergence becomes crucial: The interactions and local rules with limited local knowledge of the simple swarm members lead to an emergent global behavior through collective interaction [333].

In multi-agent systems, it is assumed that each agent can accomplish the task independently. It (i) simply takes longer if we only use one instead of many agents, or (ii) only a portion of the global goal will be achieved due to the agent’s heterogeneity or specialization. Thus, the individual agent has the ability to reach the global goal without necessarily relying on emergent behavior.

Distributed Operation

Another key characteristic of swarm systems is the lack of a central control that leads to emergent behavior. Due to the simple and local rules and the

interactions among the agents, a swarm system is, per se, a distributed system with distributed knowledge [333].

For a multi-agent system, distributed processing is not mandatory. The agents can also be controlled centrally such that parts of the task are fulfilled in a distributed manner.

1.2 Motivation to Engineer Swarm Intelligence for CPSs

With the increasing number of interconnected CPSs, individual and hierarchical control becomes impossible. The latest Study of Juniper Research in 2018 suggests that the quantity of interconnected devices will increase steadily and reach 50 billion by the year 2022¹. In contrast to static environments, the growing interconnectivity of components in modern and future technical systems has increased the complexity of system design and operation in dynamic environments. This necessitates greater adaptability, flexibility, and robustness. The systems that integrate multiple interacting components exist at various scales, ranging from System on Chip (SoC) that form the computational core of ubiquitous modern devices like smartphones, to Internet of Things (IoT), which links billions of edge devices, and the Internet of Everything (IoE), which encompasses people, processes, data, and devices. In essence, the world is transforming into a truly interconnected and collective realm, characterized by the proliferation of systems that combine numerous interacting components at different scales [333].

In this respect, CPSs have emerged as a leading domain for exploring and implementing multi-component systems, characterized by a strong interconnection between computational (software-side) and physical (hardware-side) resources [224]. The National Science Foundation defines CPSs as “Engineered systems that are built from, and depend upon, the seamless integration of computation and physical components.”. They integrate various domains, such as sensing, computation, control, and networking, into physical objects and their infrastructure, enabling interaction [123]. Thus, CPSs represent a major paradigm in the framework of collective and connected systems, as well as a vertical study of systems: They are inherently transdisciplinary, generalizing and expanding individual sub-fields such as embedded systems, robotics, and networking, by simultaneously merging concepts from cybernetics, mechatronics, design, and process science [226, 333]. Additionally, aspects related to decision autonomy, system integration, cyber-security, control, scalability, optimization, validation, and verification, play a major role in CPSs’ design and control. Furthermore, CPSs function within the physical world, which is characterized by constant dynamic changes, unpredictable events, and external conditions that are difficult to

¹Study of Juniper Research, <https://www.juniperresearch.com/press/press-releases/iot-connections-to-grow-140-tohit-50-billion>, June 2018.

model, and the involvement of other CPSs and human agents [333]. Hence, the CPS model is well-suited to describe and reason on the plethora of complex and interacting components that are being deployed, connected, and integrated with our everyday lives. This results in a multitude of interconnected and interacting components that are best characterized by the concept of Cyber-Physical System of Systems (CPSoS), which refers to a vast and distributed complex system of CPSs. In essence, this creates an ecosystem of CPSs that operate at multiple scales and interconnect with one another [274]. In a CPSoS² model, the design and control challenges become even more complex as individual component autonomy must be integrated with explicit considerations for interdependence and coordination, interoperability, distributed control, and emergence of behaviors. This high complexity necessitates methods that should achieve the following goals:

- Distributed control, supervision and management.
- Local coordination among the composing subsystems.
- Partial autonomy of the subsystems.
- Capability of dynamic reconfiguration of the system as a whole on different time scales.
- Evolution of the overall system during its operation.
- Possibility of generating useful emerging behaviors at the system level [104].

From the above-mentioned list of goals and characteristics for CPSs, it is evident that they align with the concept of a swarm system and the typical features expected in a swarm-intelligent designed system. Applying swarm intelligence to CPSs is not an entirely new idea. Significant progress has been made in the swarm robotics domain in this regard, particularly in recent years [153, 332]. However, applying the concepts of swarm intelligence to more general CPSs, particularly to real-world CPSs that are large, heterogeneous, multi-scale, and autonomous, remains a challenge yet to be fully mastered.

For the successful design and development of a swarm intelligence solution for CPS swarms, it is important to consider both the physical and cyber aspects of the involved CPSs. The optimization, networking, and physical embodiment aspects of these systems require the seamless integration of software agents, mechatronic devices, and communications, unifying them into a cohesive whole [224, 226]. A CPS can be modeled as a swarm consisting of multiple components, with each component

- integrating one or more physical devices (sensors, actuators, communication, memory, processors, etc.),

²Given that a CPSoS is a CPS itself, in the following, we will use the acronym CPS to refer in a general sense to either a single CPS or to a CPS swarm to make a distinction, if relevant.

- acting autonomously (i.e., control is distributed and/or decentralized),
- responding and possibly adapting to changing conditions, and
- locally communicating and interacting with other swarm components to possibly produce effective and useful behaviors at the system level [333].

Nevertheless, two fundamental problems can be identified in the design and deployment of CPS swarms [333]:

1. Formalization of CPSs from a swarm intelligence perspective: This process involves extracting and generalizing properties and parameters that are shared among various CPSs, which are significant for designing swarm-intelligent behavior.
2. Open research topics of the swarm intelligence domain itself (see Chapter 7 for more details).

A good example of a CPS is the consideration of an autonomous car as done in Schranz et al. [333]: At a smaller scale, each of the major components of the car (e.g., the anti-blocking system) can be described as CPSs. At a larger scale, a fleet of autonomous cars interacts and communicate with each other. This can be used as a representation of a CPS swarm. To obtain real-time sensor and traffic data, the cars in the swarm will need to communicate with other CPSs that are present in the surrounding environment and the supportive infrastructure. These CPSs can constitute an additional swarm that facilitates interaction between different swarms of CPSs. Since there is no specific limit to the number of devices or swarms, it is possible to create a large-scale, multi-level swarm system that incorporates hierarchies where required (e.g., to a central monitoring entity), integrating many CPSs that interact with multiple CPS swarms.

Chapter 2

Principles of Swarm Intelligence

A swarm system belongs to the family of self-organizing systems. Swarm intelligence is derived from the natural behavior exhibited by social animals. Their behavior tends to be adaptive, robust, and scalable [52]. These are desirable properties of an autonomous system that can be replicated with swarm intelligence approaches in the design of real-world technical systems. All three properties add additional behavioral patterns to systems that use swarm intelligence (see Figure 2.1). Additionally, a swarm is characterized by a critical number of members, micro rules and macro plans, positive and negative feedback, interactions through direct or indirect communication leading to emergent behavior, super linear effects and the ability to balance exploration and exploitation.

In summary, a group or collection of individuals can be classified as a swarm when it displays swarm behavior that includes all the aforementioned properties and characteristics.

2.1 Micro and Macro Level

In a swarm-intelligent system, local (micro) level changes drive the global (macro) level behavior through bottom-up causation. This allows agents to function in a coordinated manner without the need for central control.

There are two main approaches for modeling a self-organizing system, following the micro-level (also known as local or microscopic) and the macro-level (also known as global or macroscopic) perspective [164]. On the micro level, each system agent has its own local state space and local behavior. Hence, each agent is explicitly represented. Macro-level modeling does not look inside each agent but has a global perspective. Thus, the macro level

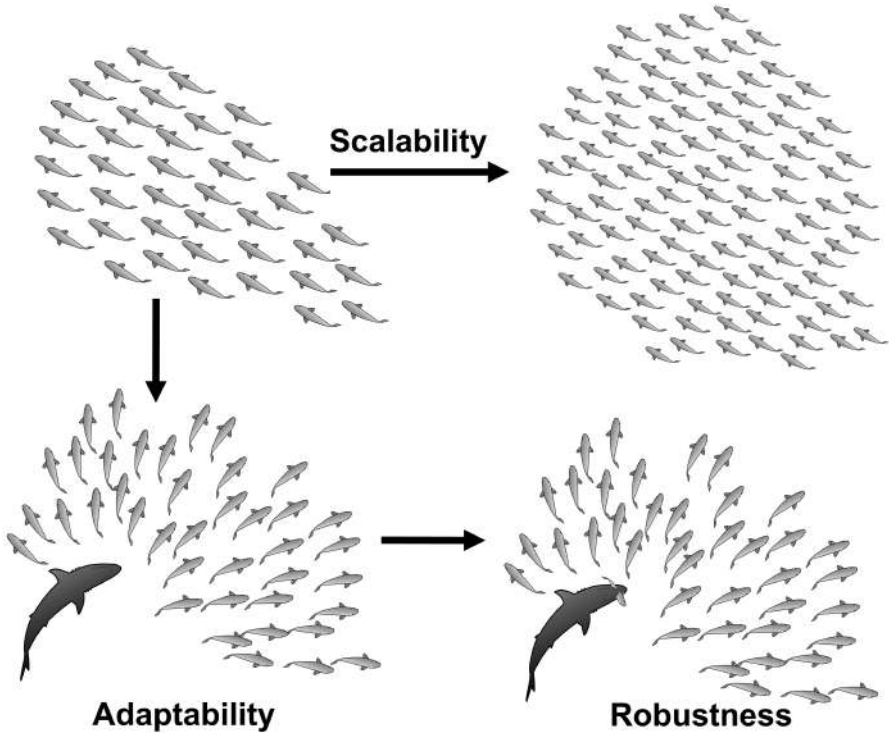


Figure 2.1: The advantages of a swarm: Adaptivity, robustness, and scalability (adapted from [292]). ↵

only considers the global state of interest and how this state changes over time. Detailed knowledge of an agent's operation is thus not required for the macro level. It is more important to model the level of abstraction to work on a global task level. Thus, the macro level abstracts agents' details away, e.g., their position or the state in the sequence of action.

Especially for modeling swarms of CPSs, the distinction between the micro and macro levels plays a crucial but contrasting role. The overall goal is defined at the macro level, involving the operation of multiple swarm agents, while each agent follows behavior defined at the micro level.

2.2 Emergence

In the design of a technical system the main question arises with regard to predictability of the system's behavior. Additionally, predictability is related to controllability that affects the ability to control the system and achieve a desired configuration and/or output. Due to the property of emergence, predictability and controllability are properties that are inherently hard to achieve in swarm-intelligent systems that follow a bottom-up approach. Thus,

emergence is directly correlated with micro and macro behaviors. Emergence is generated with local decisions and local interactions on a micro level that lead to a system function on the macro level. The most important characteristic is that this emergence can produce completely new properties that were unimaginable when considering only the micro or the macro level [153]. This aligns with Aristotle's principle that "the whole is greater than the sum of its parts". Thus, understanding the individual components of a self-organizing system is only part of the solution. While these components can be simple and easy to comprehend it is their interplay that leads to emergence, creating entirely new solutions [32].

In swarm systems, predictability and controllability are not the only challenges; any parameter or configuration setting can significantly impact the resulting dynamics of a swarm-intelligent system.

2.3 Feedback

Interactions between agents are characterized by either synergy (cooperation or positive sum) or friction (conflict or negative sum) [161, 140, 38], also known as feedback loops in complex systems theory [273]. These feedback loops lead to non-linear dynamics of swarm systems, thereby creating complex behavior.

Agents are assumed to be goal-directed: They try to maximize their fitness, satisfaction or utility. When both agents gain fitness, the interaction is said to be synergetic. When both lose fitness, the interaction is characterized by friction. When one gains while the other loses, the interaction is competitive. If the losses match the gains, it is considered zero-sum. Since agents seek to maximize their fitness, they prefer synergetic interactions, and try to avoid friction. However, since interactions are local, agents do not a priori know which interactions with other agents will be most beneficial for the overall behavior. They can discover this through trial-and-error, which mirrors the evolutionary dynamics of blind variation and natural selection of the fittest (most synergetic) interactions.

Nevertheless, friction is not always a negative characteristic. Positive feedback typically increases fitness, or other parameters that could lead to permanent growth [153]. Still, at a certain point of time increasing values do not produce any growth, but rather start disturbing each other as the resources are finite. This is where negative feedback has its positive role that stabilizes the systems final state. This again could lead to local maxima or minima that necessitate monitoring a self-organizing system's behavior.

2.4 Scalability

Swarm intelligence algorithms are able to produce complex and scalable swarm behaviors from simple and local rules, thus, on a micro-level. Scalability is a crucial aspect of a swarm system, enabling it to perform effectively with

varying numbers of swarm members and problem sizes. As noted in [85], a swarm is able to maintain its function and interaction among its parts even as its size increases, without the need for a redefinition of its interaction mechanism. This means that the addition or removal of swarm members will not result in a significant decline in performance, as long as the number of members does not fall below a certain critical mass¹ [333]. One cornerstone of implementing scalability is to restrict the agents' communication to the interaction with their local neighborhood instead of to all agents. In contrast, an all-to-all broadcasting mechanism will likely break scalability due to communication demands increasing superlinearly with swarm size.

2.5 Adaptivity

Adaptivity represents the ability of a swarm to adapt to dynamic environments, to cope with different tasks that could also appear unforeseen, and to still fulfill its mission and reach the macro-level goal. Thus, an adaptive system uses a sequence of operators applied through an adaptive plan to produce a performance that is tailored to and observable within the environment. These sequences produce different system responses dependent on the environment. Thus, the selection of the right, applicable sequence of operators is the task of the adaptive plan that defines the performance, i.e., the fitness from the operations on the environment [163].

2.6 Robustness

The vision of autonomous systems is to design a robust system behavior that can cope with a variety of unforeseeable errors and perturbations without human supervision. The reason for this is a paradigm shift from monolithic systems to large networked systems with many independent and constantly changing components. Robustness is the ability of a swarm to adapt to variations in swarm size, environmental changes or other disturbances affecting the system or the environment. Redundancy is a mechanism through which biological systems adapt to their environments [371]. In technical systems, redundancy involves cost considerations and, for static systems, management challenges.

The term robustness has different meanings dependent on the community in which it is used [221]. In the swarm robotics community, robustness is the ability of a system to cope with erroneous input², system errors, disturbances or attacks affecting specific agents or subsystems during execution [114]. This leads to other advantages, for instance, the dependability of a system.

¹It should be noted that it is not yet clarified what this critical threshold (the minimum number of swarm members) should be [153].

²1990. IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990 defines robustness as "The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions."

Generally, in a distributed system errors and small variations in input do not affect the correctness of the system's execution process. Moreover, the system as a whole exhibits graceful degradation: When arbitrary agents or links are removed or damaged, the quality of the output is likely to deteriorate, but very gradually. Thus, damaging, subverting or removing an agent may still leave the overall function intact. This is what produces robustness. This is not the case in a centralized system's control, where any malfunction of the controller represents a possible single point of failure.

Up to now, robustness is more or less a “descriptive” characteristic, without formally defined measures. As we can see in the following contributions by different authors, there is no formally defined technique that ensures the robustness of a swarm with respect to either global or local behavior.

2.7 Communication

Autonomous agents in a swarm need to continuously exchange information. Even a single byte, such as an agent's velocity or relative position, is essential for maintaining autonomy in a swarm system.

Communication in a swarm system can occur both directly and indirectly. In the direct communication an agent uses a communication medium such as Wi-Fi or infrared to transmit information to its immediate neighbor. This can be a one to one message to a single neighbor or a one to many message to several surrounding neighbors. Indirect communication involves an agent generating a message and sharing it with the environment, for example, through an access point, allowing other agents to collect the message either randomly or intentionally. The shared communication medium might be a specific location where agents deposit and retrieve messages. Indirect communication methods like depositing pheromone values, are commonly used by social insects and other animals for survival, whereas direct communication through message exchange is prevalent in the robotics domain.

Pheromone-based communication is one of the most prevalent indirect communication methods used by social animals [411]. Pheromones are complex chemical substances that facilitate intraspecific communication by being deposited into the environment by an agent and detected by others. These chemicals convey intricate messages that trigger behavioral and developmental changes within a swarm. This form of communication is widely adopted by various animals ranging from yeast and small insects like ants and bees to mammals such as dogs and humans [120, 294]. In swarm systems pheromone communication has been simulated using substances like alcohol [125], phosphorescent paint [240], light [133, 18], and infrared [285, 391]. Figure 2.2 illustrates a bio-inspired artificial pheromone-based system that mimics ants pheromone-based communication.

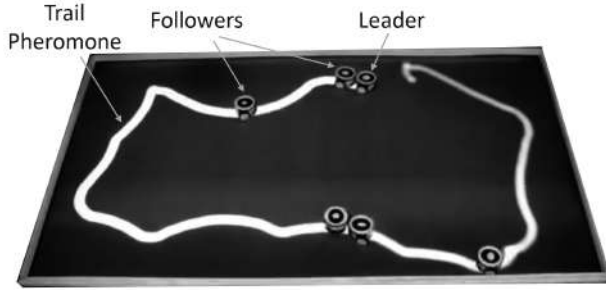


Figure 2.2: Example of an artificial pheromone system, COSΦ [18], that was developed for swarm robotic applications. In this scenario, a leader is releasing pheromones and five followers are following the leader. The pheromone trail is evaporating over time. ↻

2.8 Superlinearity

We anticipate enhanced performance in a swarm due to synergy effects and other factors including:

- **Synergy effects:** When agents collaborate they can achieve more than the sum of their individual capabilities. Their interactions can lead to innovative solutions that isolated agents could not achieve.
- **Task specialization:** Agents in a swarm can specialize in specific tasks performing them more efficiently allowing the swarm to benefit from diverse expertise and skills.
- **Parallel processing:** Tasks can be distributed among agents, enabling them to work simultaneously. This can significantly reduce the time needed to complete complex tasks.
- **Adaptive behavior:** Swarms can adapt to changing environments or tasks dynamically, optimizing their performance over time based on feedback and learning.

If the performance of a single agent is given as an execution time T_1 to complete the task, we would expect multiple agents to perform the task faster. In the following, we define performance Pf as the reciprocal value of the execution time of a task:

$$Pf = \frac{1}{T}$$

In a perfectly scaling architecture, we would expect that n agents are able to fulfill a task in $T_n = \frac{T_1}{n}$ time.

There are cases where the collaboration of agents provides a synergistic effect. In such cases, the performance even exceeds linear scaling, so that

$T_n < \frac{T_1}{n}$. An example of such a task is a scenario in which a swarm efficiently solves a complex puzzle through collective problem-solving. Each agent contributes to discovering different parts of the solution, significantly speeding up the process. A more dynamic, though admittedly ferocious, example is a hunting party tracking elusive prey. A single agent, or in this case, a hunter, would be very inefficient since the prey could easily elude the hunter. However, a coordinated group of hunters would be significantly more efficient.

In an extreme case, the performance of a single agent could be even zero while being a positive number for the swarm. Consider a scenario where robots need to clean an area by pushing obstacles out of the way. If a single robot encounters an obstacle that is too heavy to move on its own, it won't be able to complete its task. However, with the cooperation of multiple robots these obstacles can be successfully pushed aside.

Such cases are considered to exhibit *superlinear performance*. Increasing the number of agents in a group is anticipated to result in the group accomplishing more work compared to a smaller group under the same conditions and timeframe. In a group exhibiting real swarm-intelligent behavior, such a superlinear characteristic is highly desired. This characteristic indicates that the effect of the overall system is greater than the sum of the effects of its individual parts. The effective design of swarm-intelligent behavior relies on the synergies of cooperation among individual swarm members. Only when these interactions enhance the performance of each member can the overall system be considered well-designed. This means that within the bounds of a feasible swarm size, not only the efficiency of the whole swarm but also the efficiency of each individual swarm member has to increase. We refer to this as the *swarm effect*,

$$\frac{Pf_{\text{large_swarm}}}{\text{size}_{\text{large_swarm}}} > \frac{Pf_{\text{small_swarm}}}{\text{size}_{\text{small_swarm}}}, \quad (2.1)$$

where Pf_i represents any quantitative performance metric of the swarm i in which larger metric values identify a higher or better swarm performance and size_i represents the size of that swarm, given by the number of swarm members. As indicated in the formula, the average performance per agent increases with the swarm size provided the swarm is performing effectively. This principle is shown in Figure 2.3 that illustrates the expected performance scaling properties of three different systems:

- (a) The performance of an imaginary algorithm or swarm model that scales linearly with $O(n)$ as the dashed line in Figure 2.3, where n is the number of swarm members, without considering physical constraints or limitations, is not practical in the real world.
- (b) The performance scaling of an algorithm that accounts for sharing resources with other algorithm instances or a swarm model that views

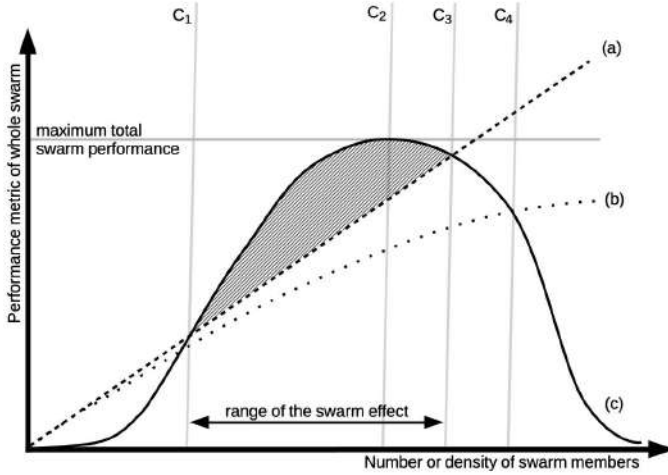


Figure 2.3: The swarm performance illustrated as a function of the number of swarm members, extracted from various experiments [37, 153]. Line (a) shows the hypothetically linear scale of a disembodied algorithm with swarm size. Line (b) presents the logarithmic scaling of a disembodied algorithm that requires shared resources among its instances when implemented for a parallel execution. Finally, line (c) shows the typical scaling of physically embodied swarms (or of physics-considering simulation of such agents) [333]. ↵

space as a shared resource for agents will scale with $O(\log n)$ as the dotted line in Figure 2.3, or even slower, due to the increased communication and coordination overhead that comes with a larger number of members.

- (c) The solid hat-shaped curve depicted in Figure 2.3 represents the performance of a physically embodied swarm system operating in the real world.

With a very low swarm size or density of agents $N < C_1$, the swarm shows almost no performance. This can be traced to the low connectivity between the agents that is not high enough to allow for cooperation and interaction. As the size of the swarm increases, the performance of the system improves due to the swarm effect, where interactions between members lead to greater efficiency. Once the swarm reaches a size of $N > C_1$, it can take advantage of these interactions to scale in a super-linear fashion. This high level of performance can be maintained within a certain size range of $C_1 > N \geq C_3$, and even beyond C_3 , the performance remains above a logarithmic scale. However, the overall peak performance of the swarm system is reached at a threshold of C_2 , beyond which the addition of more swarm members results in a decline in average performance due to crowding and other factors. Once the swarm

size surpasses another critical level of $N > C_4$, the performance of the system deteriorates and eventually leads to a total loss of performance.

Adding agents to perform a task together typically adds overhead due to several factors:

- **Additional communication:** As the number of agents increases, the amount of information that needs to be exchanged among them also increases. The demand for richer communication for the swarm to function effectively [243] can lead to higher latency and potential bottlenecks if the communication infrastructure is not optimized for scalability.
- **Coordination of movements:** Ensuring that all agents move and act in a coordinated fashion requires complex algorithms that can introduce delays. Misalignment in timing or direction can reduce the overall effectiveness of the swarm. Offline optimization of tasks [5] is limited due to uncertainties in environmental parameters. For example, a robotic drone could be delayed by wind conditions or unexpected obstacles.
- **Resource contention:** Multiple agents may need access to shared resources, such as data or physical space, which can lead to conflicts or inefficiencies if not managed properly using assignment algorithms [271].
- **Increased complexity in decision-making:** With more agents, the complexity of decision-making increases, as there are more possible interactions and dependencies to consider [245]. This can slow down response time and reduce overall efficiency.

Note, however, that a swarm is not always formed for performance reasons. For example, it could be acceptable for a swarm system to have a dropping performance for an increasing number of agents if other qualities are prioritized, such as:

- **Robustness and dependability:** A swarm system is less likely to fail completely if individual agents are capable of taking over the tasks of failed members, ensuring continued operation.
- **Maintainability:** With a modular design, individual components or agents can be easily exchanged or repaired without disrupting the entire system.
- **Extensibility:** If there is future demand, it is possible to enhance the system's performance by adding more agents, allowing for scalable solutions to problems as they arise.

2.9 Exploration and Exploitation

Swarm members being loosely coupled, can operate in two distinct modes. In *exploration* mode they disperse to explore the environment or, in case

of an optimization algorithm [276], to explore a solution space to gather information. Conversely, in *exploitation* mode, they utilize the gathered knowledge to maximize reward, such as acquiring food or identifying the optimal solution within the explored area.

One typically needs to strike a balance between exploration and exploitation to achieve optimal results. If the exploration part is emphasized too much, swarm members can get lost in a large search space without providing a valuable contribution. Conversely, a swarm that concentrates excessively on exploitation might overlook more lucrative areas. Exploration and exploitation do not have to follow a strict sequence; some algorithms like Ant Colony Optimization [90] include both concepts simultaneously. The problem of exploration and exploitation is also not limited to swarms, but is a common problem whenever there is uncertainty in a system that needs to be explored at a cost. A famous example for this conflict in probability theory and machine learning is given by the multi-armed bandit problem [195], where numerous slot machines, each creating rewards according to a specific, but unknown probability distribution, need to be explored and then exploited for maximizing gain.

Chapter 3

Modeling Swarms of Cyber-Physical Systems

A model of a system is a simplified representation derived from an actual (technical) system. Modeling a swarm of CPSs helps to lower the level of abstraction, streamline and formalize the technical system. We aim to identify a representation for each agent that is familiar to us and meets the microlevel requirements necessary for contributing to macro-level swarm behavior [248]. During abstraction we omit parts of the CPS to get a simplified system that is easier to understand. This is a typical task for engineers who must discern while still capturing the overall CPS mission. To enhance this understanding, we formally define the mission using logical descriptions and mathematics [153]. Concluding with a formal description of the system, we aim to understand how the self-organizing system behaves under various configurations and potentially in different environments.

In a self-organizing system γ , where $\gamma \in \Gamma$ with Γ as the configuration space, we can only observe the actual configuration at a specific, discrete time step t with $\gamma_t, \gamma_{t+1}, \gamma_{t+2}, \dots$. However, each initialization γ_0 gives another sequence of the system over time. To really understand a self-organizing system, abstraction and simplification are a must. Therefore, we consider a simpler system ϕ with a smaller configuration space $\phi \in \Phi$ with $\dim(\Phi) \ll \dim(\Gamma)$. Hence, we want to find a mapping f that allows us to map the configurations γ to the simpler, abstracted configuration ϕ leading to $f : \Gamma \mapsto \Phi$. From one time step to the other, from t to $t + 1$, we consider an update rule g on the real configuration $g : \Gamma \mapsto \Gamma$, thus $g(\gamma_t) = \gamma_{t+1}$, and an update rule h for the abstracted configuration $h : \Phi \mapsto \Phi$, thus $h(\phi_t) = \phi_{t+1}$. This is summarized in Figure 3.1. Finally, the abstraction implemented by f needs to be chosen carefully such that after the updates through g and h the correct, but abstracted model, remains the right one in the next time step.

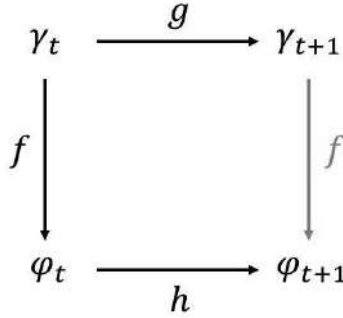


Figure 3.1: The modeling and abstraction of a self-organizing system adapted from [153]. ◀

Nevertheless, always keep in mind: “Essentially, all models are wrong, but some are useful” (George Box, 1976). In other words, all models abstract details from reality; while no model can perfectly capture its complexities, they can still provide valuable insights and guide decision-making in uncertain situations.

3.1 Modeling Principles

Generally, there is no common modeling approach for self-organizing systems or swarms of CPSs specifically [153]. Various models have been developed to work for different setups. Autonomic characteristics like self-configuration, self-optimization, self-healing, self-protecting and self-awareness schemes can be modeled by for example graph rewriting [72], L-systems [298], matrix rewriting [353], rewriting game theory [61], reaction-diffusion models [211]. In this book chapter we will not explain all the modeling approaches as they are quite diverse from application to application. We rather focus on use-case-specific models (Section 3.2 and Section 3.3) that we worked on in different projects.

When designing a model for a system, the first step is to determine which components should be represented as discrete elements and which should be modeled as continuous entities. There are four significant aspects where this decision profoundly influences the design and behavior of the model: Time, space, states of agents, and interaction. When we use a discrete micro-level model [165, 366], the behavior is usually event-driven: Each time an event occurs in the system, nodes may change their internal states and produce local output, which is sent to other nodes. This behavior can be modeled by finite automata. Since many systems are too complex to be modeled in detail, probabilities can be used to describe the behavior. This leads to the concept of stochastic automata. In a continuous micro-level model [164], the state of each node changes continuously over time. The behavior of a node can be described by a local differential equation, which takes the input values

of other nodes into account. There are numerous modeling techniques, which have already been used [248], including

- Maxwell’s demon, a model to explore the concept of entropy,
- Turing machine, a model to explore the concept of computation,
- Cellular automaton, a model for complex systems to perform discrete micro-level modeling, and
- Von Neumann’s self-reproducing automation, a model to explore the logic behind self-reproduction.

Generally, a system’s model is always influenced by its environment and the application it is intended for. This influence helps us to identify the most key features. When modeling an application as a self-organizing swarm, we must consider a set of potential swarm members and their characteristics. These agents can either be of the same type (homogeneous swarm), or different types (heterogeneous swarm). Additionally, we have established criteria to determine whether an agent is eligible to be a swarm member in the application. These criteria include whether a swarm member

- Is accompanied by a reasonable number of other swarm members in the system,
- Exhibits an appropriate level of abstraction for modeling,
- Can detect and respond to information from the local neighborhood and environment, and
- Is plausible and understandable, fostering trust during the modeling process for the proposed solution [336].

3.1.1 Challenges

When modeling a system as a swarm of CPSs several questions need to be addressed [336]:

1. What are the most representative agents and the corresponding level of abstraction?

A swarm consists of agents that interact with the environment and with each other. Each agent serves as a digital twin, an abstraction of a real CPS. An initial step in modeling a swarm algorithm requires a decision on which entity will serve as an agent within the swarm. As described in the beginning of Section 3.1, we consider several criteria to support this decision. However, determining the appropriate level of abstraction can be challenging: For instance, in a search and rescue task (see Section 3.3 for more details), we could consider robots as agents at a higher abstraction level treating robots of the same type

as agents. Alternatively, we could choose a lower abstraction level and model individual hardware components (Central Processing Unit (CPU), memory, motor blocks, etc.) as agents.

2. How to deal with inhomogeneities among agents?

Agents are inherently heterogeneous: Even minor variations in deployment can lead to differences in their reactions or behaviors, potentially resulting in unintended system outputs. For instance, in a fab setting (see Section 3.2 for more details), there is no standardized type of machine, as each can run different processes locally. Consequently, each machine has a unique set of parameters influencing its operation (e.g., furnaces use batch-processing and waiting too long to fill up the batch machine could waste time in the production process). This variability must be considered during the modeling process.

3. How to implement the necessary local swarm communication paradigms in the specific use case with the chosen agents?

Communication paradigms for swarm algorithms include direct communication and indirect communication. In direct communication agents send messages to other agents. In indirect communication, also referred to as stigmergy, agents leave information in the environment for other agents to “pick up”. Dependent on the level of abstraction the communication is performed literally among the agents. If the agents are already represented as digital twins in the central computer system, thus even more abstracted than only on CPS level, the communication can easily be implemented as local messages within the computer system or in local memory for stigmergy. The latter case also lightens the requirements for abstraction of the CPSs, as it is not necessary to equip, for instance, machines or products with additional local computational or communication intelligence.

4. How to implement a self-organizing solution on top of a working system?

Typical technical systems already include a set of mechanisms, rules, priority classes, etc., that trigger the sequence of task execution. Therefore, it is important to decide which parts of the system should run as they are, and which parts should be abstracted away—step-by-step. For example in case of priority classes, that also grow historically, the classes could be reduced to a single one by introducing swarm-based local rules and interactions.

5. How to validate the approach?

Testing and predicting the performance of a given self-organizing algorithm is difficult. Historical data that might exist for a running system, or the implementation of baseline algorithms should be used to compare with system. As the sequence of processed behavior is not deterministic, but rather stochastically differs dependent on the

initial condition and initial parameters, the validation process requires multiple simulation runs. Another challenge that arises is debugging a failed validation. While simulation runs might inform the user that the intended level of service for a given system is not achieved for some test cases, pinpointing the specific problem and rectifying it can be very challenging for a swarm system. This is because even a minor behavioral change in an agent can result in unexpected and undesired emergent behavior at the system level. Utilizing deterministic simulations with seeded random number generators can aid in the debugging process, but they do not address the overarching issue of applying a corrective solution.

3.2 Use Case: Models in the Production Plant

Scheduling in modern production plants that follow the flexible job shop principle presents a challenging, dynamic problem within the context of Industry 4.0. A typical application is the production of Integrated Circuits (ICs) in the semiconductor manufacturing industry [138]. Particularly, we consider the processing of wafers to create ICs at the so-called front end of line processing. In contrast to the high-volume production of memory ICs and CPUs, the logic and power sector manufactures fabricate specialized ICs in low-volume batches in the same plant. For this example, we consider the requirements and constraints of the leading semiconductor manufacturer Infineon Technologies AG [175]. They need to schedule between 400 and 1200 stations in their production plant producing more than 1500 different products in around 300 process steps each. The steps for a product include various processes including lithography, doping, oxidation, etching, and measuring [138].

In this use case several constraints influence the scheduling process, including the need for secondary resources, constraints from machines with equipment tooling, time couplings and batch processing. The production process involves successive steps that often loop back, requiring the same machines or machine groups to be used repeatedly. Furthermore, the production plant is required to meet various global objectives, such as maximizing machine utilization, throughput time, delivery reliability, and minimizing Work in Progress (WIP). The presence of a wide range of product diversity, coupled with the historical growth of the industrial plant, further amplifies the complexity of the system [336]. With all these constraints and requirements, we can consider the job shop scheduling problem as NP-hard [130].

Existing dispatching rules are based on heuristics and can only be used on a subset of the plant. Similarly, linear optimization methods can only cope with a subset of the plant and not with the large, and dynamic search space of an entire fab [222]. The reason is the excessive computation time needed for the calculation. Thus, these methods do not exploit the full optimization potential.

In the production plant this leads to bottlenecks that cannot be prevented, and WIP waves that will be generated. Summarized, no optimal solution for job shop scheduling has been developed so far using linear optimization, especially a solution that can be computed in polynomial time [425].

This situation paves the way for a different approach, a modeling and optimization of the production plant from the bottom-up.

3.2.1 Swarm Member Candidates

With the general problem formulation in the beginning of this Section 3.2 we identify a number of possible agents in the production plant that can act as swarm members and can be modeled in a network model [336].

Lots (L) can form swarms with lots of the same product type (homogeneous swarms), or swarms with lots of multiple product types (heterogeneous swarms). Lots follow a specific recipe through the production plant, which prescribes the order of processing steps, but not the specific machine on which to perform the next required processing step. Typically, there are multiple machines that can perform the same process. Therefore, the lots could decide which machine to prefer over the other. Additionally lots can manipulate their own priority that allows a lot to be promoted in the machine's queue. The path that is taken through the production plant by a lot is called a route. It is obvious that lots of a similar product type will share parts of their recipes. Therefore, in a stable load situation of the production plant, lots of the same and similar product types can be expected to share parts of their routes.

Machines (M) have the local information of the processes they can perform and their utilization. Given the presence of various machine types within the production plant, they can be organized into either multiple cooperating homogeneous swarms or a heterogeneous swarm with diverse capabilities. The neighborhood of each machine is determined locally and dynamically based on the recipes of the incoming and outgoing lots. Thus, the neighborhood represents not necessarily machines that are physically close, but rather those that are close according to the recipe of a lot. Machines can take decisions locally and can, if necessary, select which lot to process next by re-ordering their queue. Furthermore, they can locally communicate with other machines in their neighborhood, and could ask for lots that have their available specific processing type.

Workcenters (W), with $W \subset M$, consist of multiple related machines that could be of the same or similar type. They have attributes very similar to single machines. A workcenter can calculate when lots will be processed internally and can use the makespan information (the time between the start and the end of a lot's production) to calculate when the lots currently being processed will be finished.

Processes (P) can represent virtual swarm members. Thus, they have a view on all machines that they can potentially be run on. This also includes

the machine's current and total workload. Therefore, they can forecast the workload, the times for re-tooling at the machines, and have information on batching requirements of batch machines.

3.2.2 Network Model

For this scenario, we adopt a network model approach. The production plant is presented as a graph with the tuple $G = (V, E)$, with the notation V for the set vertices or nodes, and E for the the set of edges. We have everything that is needed for this type of modeling: Nodes that represent the machines and form a local neighborhood together with edges that indicate a dynamic path through the production plant from one node to (many) other(s). Different from the classical graph theory, nodes and edges can be added or removed dynamically, nodes can even move (if we model them as robots). Additionally, characteristics of possible topologies (e.g., star, bus, ring) are not important as these change dynamically with the nodes and edges as well.

Considering the swarm member candidates from the previous Section 3.2.1, we apply the network model leading to a network consisting of a production plant with machines, queues, processes, lots, and recipes. The production plant \mathfrak{P} consists of several sets or workcenters of machines $W^m = \{M_1^m, M_2^m, \dots\}$, where m is the machine type. Each machine M_i^m has a queue Q_i^m and every machine in a workcenter W^m can perform a process P^m . A set of lots $L = \{l_1^t, l_2^t, \dots\}$ needs to be processed in the plant, with t as the product type. Every product type t is characterized by a recipe R^t that outlines the specific sequence of processing steps required to manufacture the product. Each lot l_n^t has the flexibility to select the appropriate machines M_i^m for each necessary process step P^m from the available options.

Based on this formal representation, the recipes can be interpreted as a directed graph $G = (V, E)$ that represents the possible connections between the machines within the plant. The nodes V correspond to the machines M_i^m , and the edges E are established between two machines M_i^m and M_j^p if there exists a lot l_n^t with a recipe R^t that includes consecutive processes P^m and P^p . A route \mathfrak{R} can be defined as an ordered list of machines capable of executing the successive processes outlined in the recipe. The taken routes are a sub-graph of G with $G_{\mathfrak{r}} \subseteq G$.

3.3 Use Case: Models in Search and Rescue Applications

The SAR use case envisions a heterogeneous swarm of UAVs and Unmanned Ground Vehicles (UGVs) that is deployed in a highly dynamic disaster environment. Their task is to support first responders in real-time by locating human casualties or trapped individuals and providing first aid to those found. The use case of the acpCPS swarms can be described as follows: The

swarm of UAVs cover a defined area and collectively search for victims. As soon as a UAVs discovers a victim, it switches from the coverage to a tracking task. By tracking the victim it continuously reports the current victim's location to a specific UGV. The target UGV to be informed is selected by the UAV using a specific cost function, for instance, the distance of the UGV to the victim. Finally, the UGV is assumed to rescue the victim by navigating to the location provided by the UAV. As soon as the UGV reaches the victim, it returns to its base, and the UAV rejoins the coverage task of the UAV swarm to locate other possible victims. We implemented this use case in the CPSwarm¹ project (for more details on the project, please visit [25]).

The constellation of this mission is well suited for testing swarms of CPSs and swarm behaviors because it can benefit from several swarm characteristics (see Chapter 2 for more details): The scalability of a swarm enables the addition of more CPSs while executing a mission. An example is the case where the area to be covered is larger than expected. Robustness of a swarm guarantees a mission execution in challenging environments where individual CPSs can fail. Furthermore, unlike fully centralized control, such a swarm can continue to operate even in scenarios where connectivity is limited or sparse. The swarm functions autonomously as a self-organized system composed of diverse members, making mission-critical decisions based on the demands of the dynamic environment.

3.3.1 Model-Driven Engineering of CPS Swarms

For a use case like SAR, we need a well-defined model of the CPSs and the desired local behaviors, with scalability across different abstraction levels being crucial [225]. To design a multi-scale model, we must consider both the hardware and software components of individual CPSs, as well as the overall composition of the entire swarm of CPSs. The process of modeling a CPS swarm involves making several decisions, which encompass: (i) Deployment specifics, such as determining the number, type, and placement of CPS within the swarm. (ii) Hardware considerations for each CPS, including communication technology, interfaces, onboard sensors and actuators, processing unit, and memory. (iii) Specifying the desired local behavior of individual CPS units and the overall global behavior of the CPS swarm [345].

This leads us to a model-driven engineering process that starts from modeling the CPS hardware, their local individual behavior, and their global behavior in a swarm. Finally, this approach results in behavior code that can be deployed to and executed on the CPSs. These steps rely on a hierarchically organized set of behaviors that collectively form the controller of each individual CPS. These behaviors are based on a well-defined library of behavior models, which are linked to an automated code generation process, enabling the assembly of executable code for each CPS [345]. Based on the

¹CPSwarm website: <https://www.cpswarm.eu/>[Online; accessed 19-December-2024].

multi-scale modeling approach, it is possible to validate the swarm behavior. This is achieved by an abstract simulation allowing for rapid prototyping of the CPS behavior to observe the resulting and the desired swarm behavior. All results can be found in Sende et al. [345].

3.3.2 Hardware Modeling

When beginning to model a CPS we examine its architecture, focusing on the hardware components, their interfaces and interactions. For creating hardware models we utilize Systems Modeling Language (SysML) [15], an extension of Unified Modeling Language (UML), that focuses on modeling systems with block diagrams. To advance the modeling of CPS swarms, we enhanced the Block Definition Diagram (BDD) and Internal Block Diagram (IBD) by incorporating a CPS swarm profile [335]. It includes three types of diagrams [345]:

- **Swarm composition diagram:** We developed a swarm composition diagram that extends the BDD, enabling the modeling of the entire swarm. The diagram details the specific types and quantities of CPSs used within the swarm.
- **Hardware composition diagram:** This diagram is designed to extend the BDD that allows the modeling of individual hardware components installed utilized by the CPSs. The hardware composition diagram specifies the types of hardware components in use and defines their inputs, outputs, and other parameters.
- **Swarm member internal diagram:** We designed the swarm member internal diagram to extend the IBD to model the internal structure of a single CPS. This diagram specifies how the behaviors can interact with the environment or other CPSs by defining sensors, actuators, and communication interfaces and their parameters. On the one hand, sensors are inputs to collect information from the environment, on the other hand, actuators are outputs to interact with the environment. A communication interface, however, can provide both, inputs and outputs to the swarm behaviors and thus, enables coordination between CPSs.

3.3.3 Behavior Modeling

The behavior models describe how the individual CPS behaves, also when interacting with the environment and other CPSs in the modeled swarm. This way, the behavior models define the software components for each CPS. The individual connected behaviors are the steps each CPS must execute locally to collectively achieve the global mission that envisioned by the swarm system's designer. The design of local behaviors for the individual swarm members is a difficult task, as the emergent global swarm behavior is not easily predictable [1]. Modeling the behavior on different abstract levels supports

the process to generate the desired global behavior as it can be executed on different levels of realism and detail [345]. With this bottom-up approach, we can iteratively refine the individual local behaviors until we achieve a global swarm behavior.

Behavior State Machines

Once a complete mission is defined, we typically identify a set of distinct behaviors for each CPS in the swarm to execute, ensuring the various tasks of the mission are accomplished. These individual behaviors are treated as atomic in the modeling process, making them simple to describe and implement. By combining these simple behaviors, we can create more complex behaviors to accomplish intricate missions. A widely used yet powerful approach is to employ Finite State Machines (FSMs), where each state represents a simple behavior, and the transitions between states depict the shift from one behavior to another. Thus, the entire FSM design describes a complex behavior. During operation, each CPS runs an FSM, consistently remaining in a defined state, though this state may differ among CPSs. Hence, complex swarm behaviors can emerge where CPSs play different roles based on their interactions [345].

The transitions between the behaviors described as states in a FSM are triggered by events. These events can either originate locally, such as from sensor outputs or the behavior itself, or from the environment, like communication activities between CPSs or external commands from a global control station. The exchange of events between CPSs facilitates the coordination of local swarm behaviors. Events are processed locally and autonomously by each CPS individually as specified in their behavior FSMs. Through the exchange of events, CPSs can exert an impact on each other's behavior changes. These events are characterized by a distinct Identifier (ID), a timestamp, and a unique sender ID. Additionally, events can carry associated data, which is transmitted between behaviors.

The UML behavior FSMs [68] serve as the foundation for the behavior FSM model. Simple behaviors are represented by simple states, while composite or submachine states are used to model complex behaviors. Composite states enable the modeling of one state using another FSM, while submachine states facilitate the encapsulation of generic FSM that can be utilized in multiple states. In the realm of CPS swarm behaviors, we introduce four distinct behavior types, each denoted by a specific color code as illustrated in Figure 3.2.

- Swarm behaviors, represented by the green elements, are simplistic behaviors that execute specific swarm algorithms. These algorithms give rise to emergent swarm behaviors, such as aggregation or exploration.
- Swarm functions in blue are simple behaviors that carry out a singular function encompassing the interaction between CPSs. Examples include task allocation or the exchange of position information.

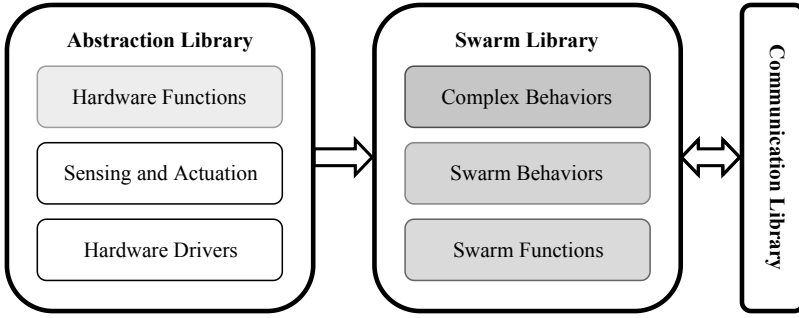


Figure 3.2: The behavior library structure [345]. ↵

- Hardware functions, depicted by the yellow elements, are straightforward behaviors that perform singular functions involving hardware interaction. Examples include moving to a designated location or controlling an actuator.

Each behavior is given a distinct name and a brief description, and is characterized by its behavior type as well as its respective inputs and outputs.

It should be noted that in related literature, such a granular differentiation between various behavior types is not commonly found. For example, swarm behaviors and swarm functions are termed collective behavior in [45] and basic swarm behavior in [337]. In [124], the term “constituent behavior” is employed to refer to any kind of simple behavior, including hardware functions. However, we intentionally introduce the distinction between behavior types to promote a structured and organized design of the behavior FSMs. This is facilitated through the use of hierarchically nested states, as outlined in the UML standard. This enables the abstraction of behavior details at higher levels. For instance, an aggregation behavior can be implemented independently of the method employed for exchanging position information between CPSs. Events can be utilized to trigger a state change within the sub FSM or cause the higher-level behavior to terminate, thus also concluding the currently running sub-behavior.

Behavior Libraries

To enable frequently recurring behaviors and functionalities to be defined only once, the behaviors are organized into libraries. The library structure we propose is illustrated in Figure 3.2, with each color corresponding to a different behavior type as previously defined. The libraries are organized based on the level of hardware abstraction, which facilitates the separation of concerns. They comprise software artifacts that are modeled as states within the FSMs. The Swarm Library is hardware-independent and contains the complex behaviors modeled as FSMs, in addition to swarm behaviors and swarm functions. It utilizes a Communication Library, which provides a

communication interface between CPSs. The Abstraction Library abstracts the hardware specifics and provides functions related to hardware, as well as functionalities to access sensors and actuators using hardware-specific drivers.

The Swarm Library consists of swarm behaviors executed by individual CPSs, resulting in the global behavior of the swarm. They are platform-independent and can be reused across different types of CPSs. The library is divided into three sub-libraries based on the previously introduced swarm behaviors. Firstly, the Complex Behaviors Library contains FSMs that model complex, high-level mission behaviors such as SAR, and are defined as UML composite or submachine states. Secondly, the Swarm Behaviors Library comprises individual swarm algorithms that exhibit emergent behavior. These swarm algorithms are typically based on biological inspiration or are generated automatically, such as through evolutionary optimization. Examples include flocking, phototaxis, or collective transport, and they are defined as UML simple states that are used in the complex behavior FSMs. Lastly, the Swarm Functions Library includes simple swarm-related tasks that do not lead to emergent behavior, but rather enable the functioning of the swarm behaviors. Examples include the exchange of position information, task allocation, or computation of the average velocity of the swarm. They are defined as UML simple states to be used in the complex behavior FSMs.

The Communication Library provides a set of communication services to the swarm, which are built on top of an arbitrary network interface. These services encompass various functionalities, such as transmitting telemetry data from the CPSs to the command and control station, facilitating the exchange of events between CPSs, as well as between CPSs and the command and control station, and enabling remote access to parameters of the CPSs within the swarm. This library was developed as part of this work and is available on GitHub [297].

The Abstraction Library provides a set of functions and interfaces that abstract away the hardware specifics of the CPSs. It allows the development of high-level routines from an application-oriented perspective, raising the level of abstraction from a platform-dependent point of view. This enables developers to focus on describing how the CPSs should behave in order to complete a high-level task or reach an application-specific goal, without being concerned with the underlying hardware implementation details. This is achieved by providing a set of CPS-specific libraries in order to access platform-specific information of a acCPS in a standard and coherent way.

The Abstraction Library is designed to provide a higher-level interface for accessing the hardware of the CPSs, allowing developers to focus on high-level tasks and goals rather than low-level hardware details. The library is composed of three layers, with each layer adding a level of hardware abstraction. The bottom layer, Hardware Drivers, includes all the drivers for sensors and

actuators that are mounted on the CPSs. The middle layer, Sensing and Actuation, is responsible for providing sensor information and controlling the CPSs using their actuators. The Hardware Drivers layer establishes a direct connection with the hardware, while the Sensing and Actuators layer employs software to offer an initial level of abstraction by implementing intricate functionalities needed by the higher layer. At the topmost layer, the Hardware Functions layer encompasses a collection of high-level functions that represent sophisticated routines executed by a CPS, involving a combination of sensors and actuators. These functions are used to define the states of the FSMs as UML simple states.

3.3.4 Code Generation

The responsibility of the code generator is to convert the FSM models into executable code that can be deployed on the CPSs. The generation process is based on templates, which are suitable for structures that are schematic and repetitive. With template-based generation, a basic set of templates is defined, which is then populated with data extracted from the algorithm specification.

The templates consist of a static part that appears in the output as-is and a dynamic part that is replaced by input data using a template meta-code. The meta-code includes directives that are processed by a template engine along with the input data to produce the final source code. To target different runtime platforms, different templates are used. The code generator uses a set of templates written in the Velocity Template Language (VTL) to generate executable code.

The code generator processes the data in the form of FSM models of complex behaviors, which are translated into State Chart XML (SCXML) files. These files contain all the necessary information for the code generator, including the type of functionality used to select the appropriate template and the definition of the Application Programming Interface (API) executed in a given state.

SMACH is a Python-based project that provides a framework for the implementation and execution of FSM-based algorithms. The code generator, which is part of this project, uses the SCXML file and a set of templates to produce Python code that implements the designed FSM. This code generator is available on GitHub [296].

3.3.5 Modeling on the Example of the SAR Use Case

In this section, we will show how the formal approaches proposed earlier can be applied to a heterogeneous swarm of CPSs. We will describe how we model the hardware and software components that were used in our experimental evaluation. Our main objective is to demonstrate the feasibility of deploying software onto CPSs based on the models created using formal methods. The diagrams were created using the modeling tool Modelio. Subsequently, the data was exported for additional processing by the code generator, utilizing

the CPSwarm Modeler module. This module can be found on GitHub [295], providing the necessary functionality for the generation of code based on the exported data.

We have selected the SAR use case to showcase the formal approaches proposed in this section. CPSs have various applications in such missions, such as creating a situational overview, providing logistic support, serving as a repeater or surrogate for other CPSs, and clearing debris, as reported in Murphy et al. [262]. To make the SAR use case more manageable for deployment on prototype platforms that do not necessitate specialized hardware, our focus is primarily on two tasks: Locating human casualties or individuals trapped in the disaster site and providing initial medical assistance. To accomplish this, we implement a heterogeneous swarm comprising both UAVs and UGVs. The mission of the heterogeneous CPSs swarm can be described as follows: The UAVs cover a designated area to search for victims, while the UGVs remain inactive. Once a victim is detected by a UAV, it switches to tracking mode and continuously tracks the victim's position, which is communicated to the UGVs. An arbitration process is initiated by the UAV to select the most suitable UGV to reach the victim, based on a cost value provided by the UGV, such as the distance to the victim. The chosen UGV navigates to the victim using the received position information from the UAV, and once the victim is rescued, the UGV returns to its starting position while the UAV resumes coverage to search for a new victim.

As an example, we will focus on modeling the UAVs, including their hardware models and behavior implementation based on Robot Operating System (ROS) [300], while omitting the simpler UGV models for brevity. To simplify the SAR use case, we will refer to victims as targets.

Hardware Models

The authors of this work have custom-made the UAV platform as prototypes. The UGV platform is based on an off-the-shelf RC truggy. The UAV prototype platform is represented by an extended SysML IBD model, as depicted in Figure 3.3. The model defines the inputs such as sonar range finders, camera, and Ultra Wideband (UWB) localization. The outputs for controlling the locomotion and a communication interface for inter-CPS communication and communication with the command and control station are also specified. Further information on this modeling process can be found in [332].

Behavior Implementation

To represent the behavior of the UAVs, we utilize a two-level FSM hierarchy denoted as $H = \{L_1, L_2\}$. This hierarchy effectively models the swarm behaviors using a collection of swarm and hardware functions sourced from various libraries. The implementation of these behavior libraries is built upon ROS, an open-source framework specifically designed to simplify the creation

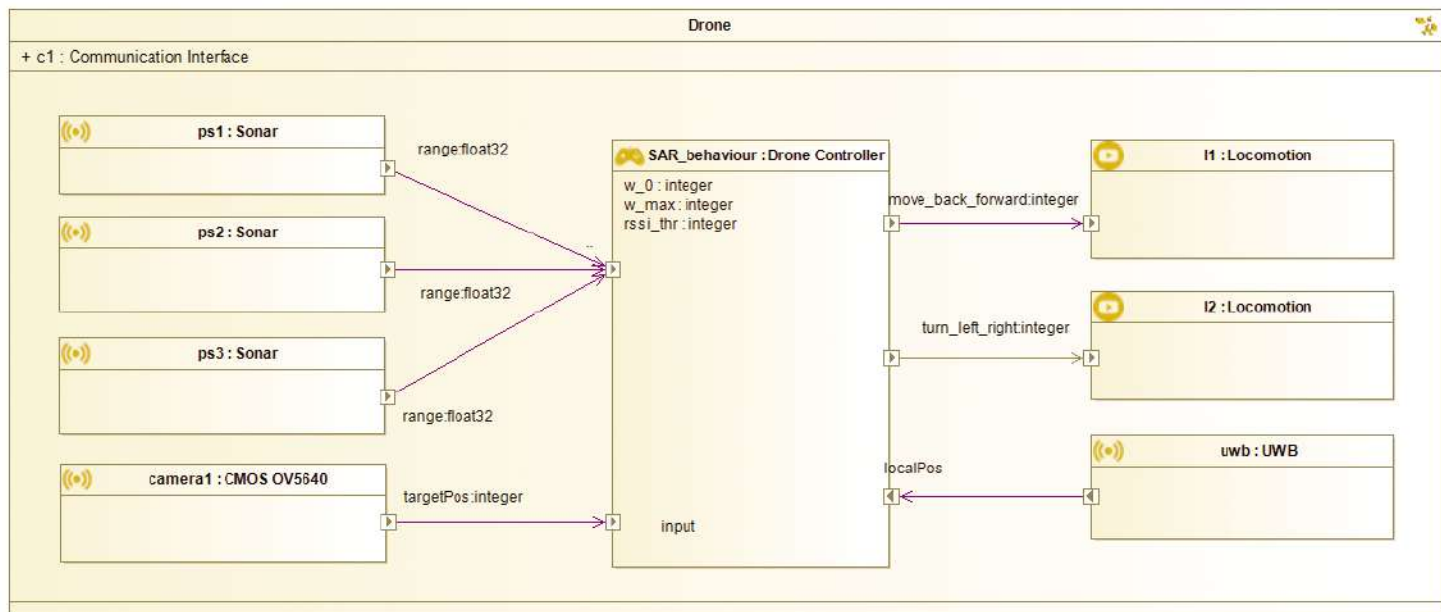


Figure 3.3: The hardware model of the UAV prototype [345]. ↩

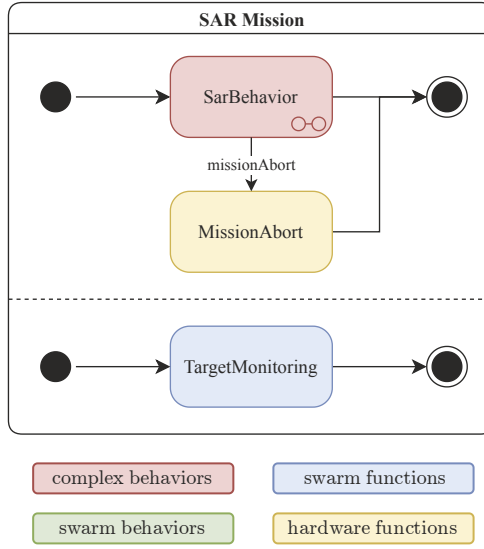


Figure 3.4: The UAV SAR mission processes [345]. ↵

of robot software. ROS has gained widespread adoption in the robotics community due to its exceptional flexibility and extensive hardware platform support, making it the de facto standard in the field.

The first level L_1 in the hierarchy defines the parallel processes that are executed on each UAV, as depicted in Figure 3.4. The *SarBehavior* state models the actual SAR behavior required to complete the mission and is itself a complex behavior modeled as a L_2 FSM. The *missionAbort* event allows for the immediate termination of the *SarBehavior*, along with any currently running behavior in the corresponding L_2 FSM. Therefore, an event broadcasted by the command and control station to all CPSs can stop the entire mission. There are various swarm and hardware functions executed in parallel, which are specified in detail in Tables 3.1 and 3.2, respectively.

The UAV FSM hierarchy L_2 consists of the second level, which models the behavior of the UAV during the SAR mission. It is depicted in Figure 3.5. When a UAV is powered on, it starts in the *Idle* state. Upon receiving the *launch* event from the command and control station, the UAV transitions to the *TakeOff* state and ascends to the designated altitude. Once the desired altitude is reached, the UAV enters the *Loitering* state and remains stationary. The actual SAR mission starts when the *missionStart* event is received. The coverage state is the initial state of the SAR behavior, during which the UAV searches for targets. If a target is detected, the *TargetMonitoring* state triggers the *targetFound* event, leading the UAV to the *SelectRover* state. During this state, the UAV communicates with all available UGVs and selects the closest one to move towards the target. As the selected UGV moves towards the target, the UAV performs the *Tracking* behavior to keep track of the target's

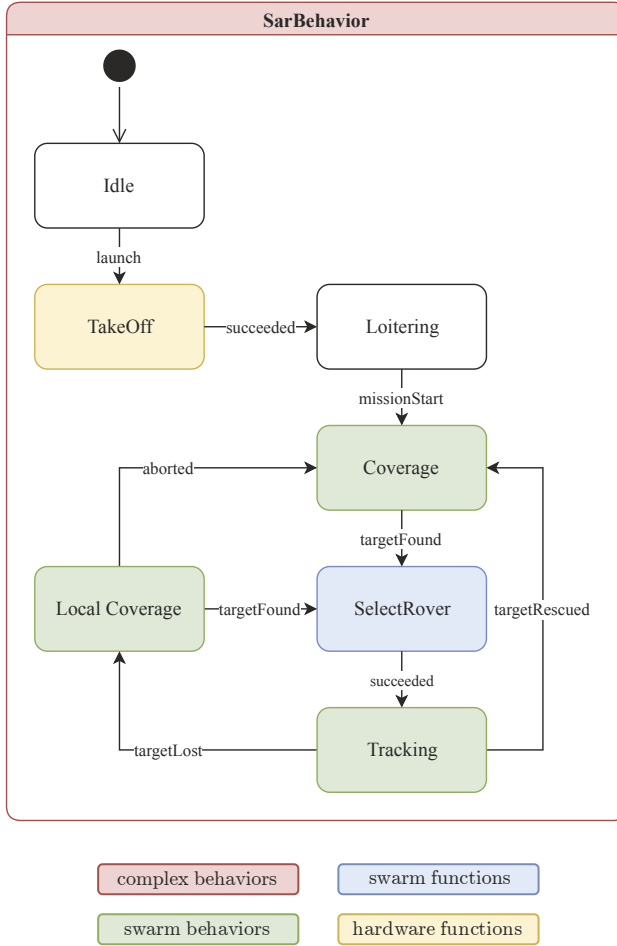


Figure 3.5: The complex SAR behavior of the UAVs [345]. ↩

location until it is reached by the UGV. It informs the UGV about changes in position through the *targetUpdate* event. If the *targetRescued* event is received from the UGV, the target is safe, and the UAV restarts the “Coverage” behavior to search for other targets. If the target is lost before being rescued, the UAV enters the LocalCoverage state, which allows it to circle around the last known target position to find the target again. If the target is found, the UAV selects the most suitable UGV again. If the target cannot be found, the UAV restarts the “Coverage” behavior.

The SAR mission’s FSMs utilize behaviors and functions from the Swarm Library and the Abstraction Library. Table 3.3 and Table 3.1 present the swarm behaviors and functions used in the SAR mission, respectively. The complete libraries are available as ROS stacks, the behaviors [341] as well as the functions [342], and their details can be found in the links provided in

Table 3.1: The swarm functions used in the SAR mission as part of the Swarm Library [345]. ↵

Behavior	Input	Output	Description
TargetMonitoring	camera footage	IDs of targets	Manages targets being detected by the swarm. It uses the on-board camera to detect targets and exchanges information about targets with the other CPSs in the swarm.
SelectRover	target ID, target position	UGV ID, target ID, target position	Assigns the closest idle UGV for rescuing a target using the <i>targetAssigned</i> event.

Table 3.2: The hardware functions used in the SAR mission as part of the Abstraction Library [345]. ↵

Behavior	Input	Output	Description
MissionAbort	-	-	Lets a UAV land.
TakeOff	altitude	-	Lets a UAV lift off.

Table 3.3: The swarm behaviors used in the SAR mission as part of the Swarm Library [345]. ↵

Behavior	Input	Output	Description
Coverage	coverage area boundaries	target ID, target position	Lets a UAV fly over the mission area in search of targets. It terminates once it finds a target.
Tracking	target ID	target position	Lets a UAV keep track of a target that has been found. It informs other CPSs about position changes using the <i>targetUpdate</i> event.
LocalCoverage	last known target position	target ID, target position	Lets a UAV search the local neighborhood for a lost target. This exploits the fact that a target has a high probability of still being close to the current position of the UAV.

the footnotes. The hardware functions of the Abstraction Library utilized in the SAR mission are listed in Table 3.2. There are other modules pertaining to different levels of the Abstraction Library released on GitHub. For more information, please refer to the repositories of the Hardware Functions [343] and Sensing and Actuation [344] libraries, as specified in the footnotes. The events that trigger state changes in the behavior are summarized in Table 3.4.

Table 3.4: Events used in the SAR mission [345]. ↵

Identifier	Data	Sender
launch	-	command and control station
missionStart	-	command and control station
missionAbort	-	command and control station
targetFound	target ID, target position	swarm member
targetUpdate	target ID, target position	swarm member
targetLost	target ID, target position	swarm member
targetRescued	target ID	swarm member
targetAssigned	target ID, UGV ID	swarm member

3.4 Use Case: Models in Edge Computing

The shift toward local processing at the edge brings critical benefits that address future computing challenges, including enhanced security, improved reliability, reduced latency, and lower energy consumption. Managing this edge infrastructure, often called the edge continuum, creates a dynamic and adaptable computing environment.

In this use case, we examine a network of Edge Micro Data Centers (EMDCs) within the edge continuum (see Figure 3.6), where intelligence is distributed across the nodes, creating a decentralized environment. This distribution enhances the edge’s autonomy and granularity in local decision-making within a regional context, reducing dependence on a central coordination point. This is particularly crucial for real-time applications such as autonomous driving or the monitoring and control of smart grids. The stability and performance of edge infrastructure are increasingly challenged by stringent requirements for latency and autonomy, distribution across multiple locations, limited local size, multi-tenancy, the involvement of multiple operators, and local management, with components operating concurrently and asynchronously.

These challenges are further intensified by the rapid growth of i) connected devices and their data exchange capabilities, ii) intelligence embedded in edge devices, iii) the breakdown of monolithic applications into smaller components, and iv) the scale, speed, and complexity of interactions among devices in a zero-trust environment. As a result, orchestrating edge and cloud interactions, particularly in resource allocation, workload scheduling, and data management, is becoming increasingly complex (see Figure 3.6 for details on the architecture [334]).

To tackle these challenges, we propose an approach centered on autopoietic systems—self-organizing, self-regulating, and self-repairing. AI-driven optimization methods, such as swarm intelligence, have been successfully applied in cloud environments and are central to our design. Our framework uses swarm agents to represent demand and supply entities. Demand swarm agents optimize operations at the pod level, while supply swarm agents manage tasks such as workload placement and caching within

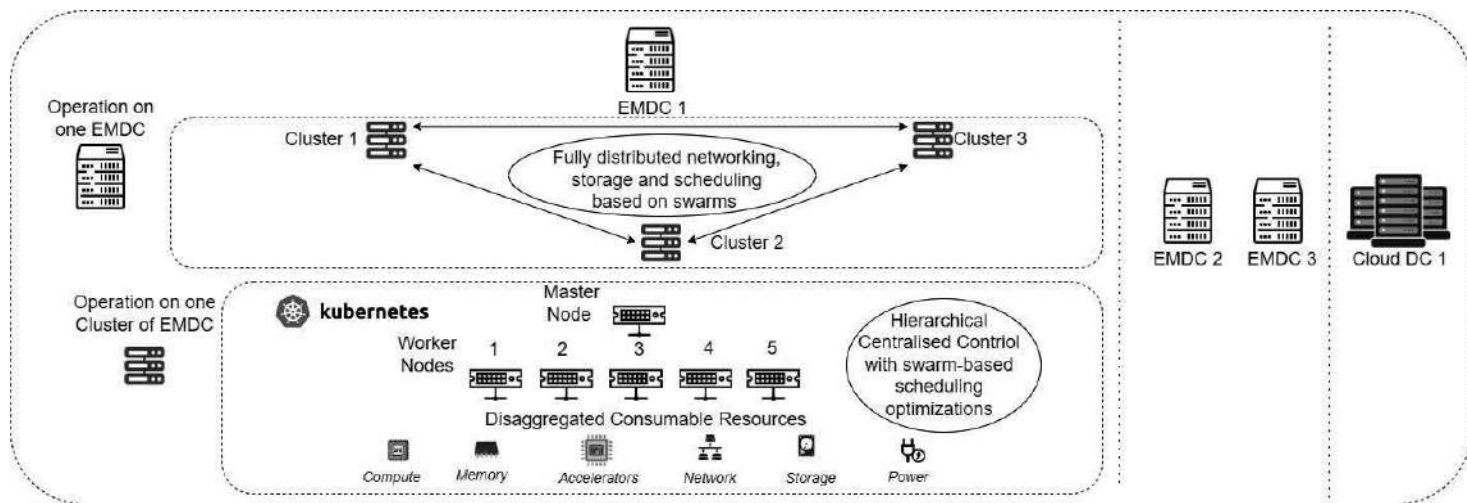


Figure 3.6: Schematic architecture demonstrating the inter-edge resource allocation in clusters: Nodes, pool of resources and the overall edge-cloud interaction [334]. ↵

EMDCs. Communication between these agents is facilitated by synthetic hormones, with supply agents detecting these signals to make informed resource allocation decisions. Inspired by ant foraging behavior, the ant algorithm is applied to optimize workload-node assignments, using pheromone trails to guide future decisions. For further information, refer to Schranz et al. [334].

3.4.1 Modeling Agents in the Edge Continuum

Our agent-based approach employs two distinct types of swarm agents: Demand and supply swarm agents. These agents work collaboratively within an EMDC environment, managing tasks such as pod placement, storage management, and caching optimization. The model represents an edge continuum comprising resources, queues, pods, and processes.

Demand Swarm Agents

Applications are decomposed into a set of services \mathfrak{S} , each represented by a collection of related pods $P^s = \{p_1^s, p_2^s, \dots\}$, where s denotes a specific service. Each service s is characterized by a set of resources R^s , detailing the processing steps required to execute the individual pods. Each pod p_j^s has the flexibility to select from suitable nodes N_i^n to execute the necessary process steps P^r .

Demand swarm agents operate at the pod level within the EMDC architecture, equipped with a pod-level description that contains contextual information about the workload, such as workload type, performance requirements, data dependencies, and latency constraints. These agents leverage this information to make intelligent decisions regarding workload placement and resource allocation, enhancing the overall efficiency of the system.

Supply Swarm Agents

The EMDC \mathfrak{E} encompasses various sets of nodes, each with different resource types $N^r = \{N_1^r, N_2^r, \dots\}$, where r represents the resource type(s). A typical EMDC node may offer multiple resource types, whereas a node with a single resource type might represent, for example, a CPU within a larger resource pool. In this context, we consider resources such as CPU, FPGA, RAM, and NVMe, each with specific capacities and an associated queue Q_i^r .

Supply swarm agents operate at the node level within the EMDC, representing individual nodes characterized by their available resources and respective capacities. These agents play a crucial role in the efficient allocation of resources to incoming workloads, ensuring optimal processing and resource utilization.

Agent Collaboration and Self-Organization

Collaboration between demand and supply swarm agents is facilitated through swarm intelligence algorithms. Demand swarm agents autonomously identify the most suitable nodes for workload placement, while supply swarm agents select the optimal workloads to process based on their available resources and capacities. This collaborative, self-organizing decision-making process allows the system to dynamically allocate workloads to nodes, optimizing processing performance, reducing latency, and maximizing resource utilization.

Our agent-based model is designed to exhibit autopoietic characteristics, promoting self-organization, regeneration, and regulation within the edge continuum. Through continuous interaction and adaptation to varying workloads and resource conditions, demand and supply agents foster emergent behaviors that enhance the system’s resilience and operational efficiency.

3.4.2 Challenges in Modeling Agents for EMDCs

Agent-based modeling of an EMDC presents several challenges that must be addressed during the modeling process.

Pool of Resources

Beyond the nodes in an EMDC, we consider a pool of resources as an innovative addition to the current definitions of the edge continuum. This enhancement allows for the allocation of individual resources for pod processing, separate from the traditional node-based processing capabilities, which consist of multiple resources. The resource pool is integrated into the EMDC and can be accessed by edge or edge-cloud management as needed. This approach helps prevent resource limitations, minimizes latency, and ensures the stability of other pods’ performance by protecting their assigned resources from being exhausted. Currently, technologies like Compute Express Link (CXL) are being integrated into CPUs (by Intel and AMD), memory, and storage components (e.g., by Samsung), with PCIe switches anticipated by 2025. Aside from hardware development, the primary challenge lies in orchestrating these resource pools effectively. Current hub-and-spoke orchestration mechanisms are insufficient for managing such ad hoc configurability.

Application Types

Different services correspond to various application types, each with unique response time requirements:

- **Long-Running Applications (LRAs):** These applications instantiate long-standing pods to support iterative computations in memory or continuous request-response cycles. Examples include processing frameworks like Storm [367], Flink [121], and Kafka Streams [189]; latency-

sensitive database applications like HBase [157] and MongoDB [254]; and data-intensive in-memory computing frameworks like TensorFlow [378].

- **Batch Processing:** This method is typically employed when large volumes of data need to be processed at once, with results stored for later use. Batch processing can occur on a scheduled basis or at regular intervals and includes two types: Regularly recurring requests and opportunistic requests with minimal or no SLA (Service Level Agreement) requirements.
- **Stream Processing:** This approach handles large data volumes that require real-time processing.

Future workloads are expected to become even more complex, with LRAs, batch processes, and stream processes increasingly interconnected. This will complicate the task of categorizing applications and fine-tuning their corresponding agents.

Relationships among Pods

Demand swarm agents represent pods P^s derived from a specific service s , and these pods may have various interdependencies. Pods may need to be processed in parallel or have dependencies on one another. When a pod underperforms, the current system may generate additional pods to meet the required response times for the service s . However, these interrelationships are not currently considered in scheduler and orchestration optimization. For instance, strategically placing interacting services in proximity can significantly enhance performance:

- **Service Clustering:** For services comprising multiple microservices that frequently interact, placing microservices within the same region can improve performance.
- **Database Proximity:** Pods with heavy database dependencies should be positioned close to the database to minimize latency and enhance overall performance.

Addressing these challenges involves refining the scheduling and orchestration processes to leverage these pod relationships, thereby optimizing system performance and efficiency.

Chapter 4

Engineering Swarm Behavior

Designing behavior in CPS swarms presents a significant challenge. We've placed a strong emphasis on distinguishing between micro-level and macro-level modeling of behavior, as detailed in Section 2.1. The mission of the swarm is primarily defined at a global macro level. To accomplish this mission, we must translate local rules into executable code on controllers and processing units within each CPS, incorporating hardware-related CPS functionalities such as sensors and actuators.

4.1 Basic CPS Swarm Behavior

Figure 4.1 provides a taxonomy of swarm behaviors, based on the classification proposed by Schranz et al. [337]. In the subsequent sections, we'll start by providing a comprehensive overview of this taxonomy. For an exhaustive description of the pre-existing categories, we kindly direct interested readers to [45, 337].

Spatial Organization

These behaviors enable robots in a swarm to navigate their environment, enabling them to organize themselves or objects spatially.

- **Aggregation** causes individual robots to converge in a specific area of the environment, enhancing spatial proximity among swarm members and encouraging further interactions.
- **Dispersion** operates in contrast to aggregation, as it drives individual robots to spread across the environment. Dispersion can be

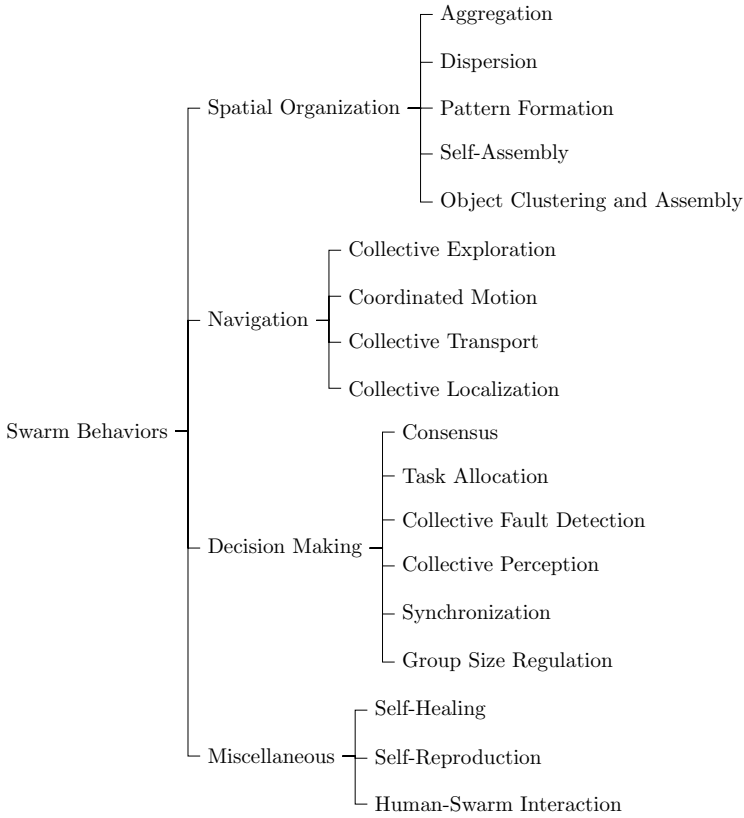


Figure 4.1: Taxonomy of swarm behaviors (extended from [337]). ↵

advantageous in scenarios where spreading out allows for more extensive coverage of the environment or helps avoid congestion and overcrowding in certain areas.

- **Pattern formation** orchestrates the arrangement of the robot swarm into specific shapes. A notable instance is chain formation, where robots align in a line, often to establish multi-hop communication between distant points.
- **Self-assembly** interconnects robots to establish structures, either through physical connections or virtual communication links. An interesting variant is morphogenesis, where the swarm autonomously transforms into a predefined shape, exhibiting remarkable adaptability.
- **Object clustering and assembly** empowers the swarm of robots to manipulate spatially distributed objects, crucial for various construction processes, allowing for the efficient clustering and assembly of objects within the environment.

Navigation

These behaviors facilitate the coordinated movement of a robot swarm within an environment:

- **Collective exploration** guides the swarm of robots to navigate cooperatively through the environment, serving various purposes such as gaining a comprehensive situational understanding, searching for specific objects, monitoring the surroundings, or establishing a robust communication network.
- **Coordinated motion** orchestrates the movement of the robot swarm in a structured formation, which can take on well-defined shapes like a straight line or adopt more flexible arrangements, such as flocking, enhancing their collective mobility and adaptability.
- **Collective transport** empowers the robot swarm to collaboratively transport objects that exceed the weight or size capabilities of individual robots. It enables the efficient movement of bulky or heavy items by leveraging the combined strength and coordination of the swarm.
- **Collective localization** facilitates the robots within the swarm in determining their precise position and orientation relative to one another by establishing a local coordinate system across the entire swarm.

Decision Making

These behaviors enable robots within a swarm to reach a unified decision on a given matter:

- **Consensus** empowers individual robots in the swarm to reach agreement or convergence towards a single, shared choice among multiple alternatives, fostering cohesive decision-making.
- **Task allocation** dynamically assigns emerging tasks to individual robots in the swarm, aiming to optimize the overall performance of the swarm system. When robots possess diverse capabilities, tasks can be distributed accordingly to enhance system efficiency.
- Within the swarm of robots, **collective fault detection** identifies deficiencies in individual robots, enabling the identification of outliers or deviations from the desired swarm behavior, often attributed to hardware failures.
- **Collective perception** combines locally sensed data from robots in the swarm into a comprehensive view, facilitating informed collective decision-making. This behavior allows the swarm to reliably classify objects, allocate the appropriate number of robots to specific tasks, or determine optimal solutions to global problems.

- **Synchronization** aligns the frequency and phase of oscillators among robots in the swarm, ensuring a shared understanding of time. This synchronization enables robots to perform actions synchronously, contributing to coordinated and time-sensitive activities.
- **Group size regulation** empowers robots in the swarm to form groups of desired sizes. If the swarm's size exceeds the specified group size, it autonomously divides into multiple groups by simultaneously maintaining optimal group cohesion and function.

Miscellaneous

There are additional behaviors exhibited by swarm robots that do not fall into the aforementioned categories:

- **Self-healing** enables the swarm to recover from faults stemming from individual robot deficiencies. The objective here is to minimize the impact of robot failures on the overall swarm, thus enhancing its reliability, resilience, and performance. This concept aligns with the collective fault detection mentioned before.
- **Self-reproduction** empowers a swarm of robots to either generate new robots or replicate the patterns created by multiple individuals within the swarm. The aim is to enhance the swarm's autonomy by eliminating the necessity for human engineers to manually create new robots.
- **Human-swarm interaction** allows humans to control the swarm robots or receive information from them. This interaction can occur remotely, such as through a computer terminal, or in close proximity within a shared environment, facilitated through visual or acoustic cues.

4.2 Use Case: Collective Motion

Collective motion in the form of a flocking behavior, is a natural phenomenon seen in bird flocks, fish schools, and molecule movement [52]. These organisms coordinate their movement to enhance performance, such as fish evading predators and maintaining group structure or birds flying in energy-efficient V-formations. Inspired by this, various algorithms have been derived from biological examples or developed newly to address challenges in swarm robotics. Overall, these studies address how large groups of simple, small, often identical robots can collectively perform complex tasks that single robots cannot. A self-organized collective motion occurs when a swarm of robots moves cohesively, using local interactions to exchange essential information like positions, velocities, or angles.

Reynolds [311] established the foundation for modeling flocking behavior in artificial systems using so-called Boids agents. The model achieves cohesion through attraction between robots and prevents collisions via repulsive

interactions, while alignment ensures the swarm moves in the same direction at the same speed. This model assumes that each robot knows its neighbors' headings in a noise-free environment. Another method, the Self-Propelled Particles (SPP) approach, uses principles from statistical physics to achieve flocking in robots. The SPP model uses virtual attractive and repulsive forces between swarm entities to achieve self-organized collective motion [395]. There are two main approaches to reach flocking behavior: The alignment rule and the position-based rule. The alignment rule involves sharing angle information among agents to agree on their headings [394], but it requires sophisticated measurement and communication tools for reliable orientation sharing. The position-based rule, which only requires agents to detect their neighbors' positions, is a more cost-effective choice. Swarm cohesiveness and alignment are achieved using just relative position data [116]. However, it should be mentioned that, although these algorithms are cost-efficient, relying solely on relative position data results in slower response times and reduced robustness.

The SPP concept was utilized in the Standard Vicsek Model [394], an early method for flocking. This model, designed to study the effects of noise and particle size on order transitions, yielded promising results with a velocity alignment rule, where each particle adjusts its trajectory to align with the average heading of its neighbors. While velocity alignment proved useful, Couzin et al. [73] introduced another model for three-dimensional environments, considering three interactions—attraction, alignment, and repulsion—in distinct zones to simulate animal group behavior transitions.

These foundational flocking models have inspired numerous studies in swarm robotics. For example, Cucker and Smale [74] proposed a method where informed agents direct a uniformed swarm using the velocity alignment rule. This model updates agents' velocities based on a Laplacian framework, regulating velocity differences with neighbors, and inspired real-world experiments [385]. However, these frameworks require exchanging both velocity and alignment data, demanding significant onboard processing, which is challenging for small robots with limited power and processing capabilities.

Several studies have focused on implementing collective motion without relying on orientation information. For example, velocity alignment is achieved implicitly through pairwise repelling forces [244]. Other approaches include using position-based attraction and repulsion [370], inelastic collisions between isotropic agents [150], and various collective control strategies [118]. These methods avoid orientation sharing and offer distinct advantages.

Therefore, reducing the information exchanged between robots can minimize hardware complexity and cost. Additionally, studies show that alignment without explicit orientation exchange lowers the swarm's energy consumption. Ferrante et al. [116, 117] developed a similar approach with the Active Elastic Sheet (AES) model, which uses elastic interaction based on relative position for collective motion. The AES model's performance is well-documented, addressing factors like network architecture [386], robustness against measurement noise [427], and behavior under external force to guide the swarm [305]. To explain the AES model in more detail, consider a swarm

system of N robots moving in a two-dimensional arena. The motion of the i^{th} robot is determined by attraction-repulsion forces from its closest neighbors. The positions \vec{x}_i and orientations θ_i of the robots can be calculated mathematically, as described in [116, 117]. The robots are deployed in an ideal environment with negligible noise in the original signals. Thus, it is possible to eliminate disturbance terms and present a modified model as shown in Equation (4.1):

$$\dot{\vec{x}}_i = (v_0 + \alpha \vec{F}_i \cdot \hat{n}_i) \hat{n}_i, \quad \dot{\theta}_i = \beta \vec{F}_i \cdot \hat{n}_i^\perp, \quad \hat{n}_i = [\cos(\theta_i) \quad \sin(\theta_i)]^T, \quad (4.1)$$

where, α and β are inverse transitional and rotational parameters, and v_0 is a biasing speed. \hat{n}_i is a unit vector parallel to the heading direction of the robot i , and \hat{n}_i^\perp is a unit vector perpendicular to it. The interactions between robot i and its neighbors s_i will generate a linear force \vec{F}_i to maintain the distance within the swarm. This force can be obtained using Equation (4.2):

$$\vec{F}_i = \sum_{j \in S_i} -\frac{k}{l_{ij}} (|\vec{r}_{ij}| - l_{ij}) \frac{\vec{r}_{ij}}{|\vec{r}_{ij}|}, \quad \psi = \frac{1}{N} \left\| \sum_{i=1}^N \hat{n}_i \right\|. \quad (4.2)$$

Here, $\frac{k}{l_{ij}}$ is the spring constant, and l_{ij} is the natural length connecting robots i and j . The distance between the i^{th} and j^{th} robots is represented as $\vec{r}_{ij} = \vec{x}_j - \vec{x}_i$. Initially, the i^{th} robot is connected to its neighboring robots S_i through virtual springs, establishing the formation of the robots at the initial stage. Thus, this spring connection remains constant despite changes in distance between robots during the experiments. The alignment of the entire swarm determines the collective flock's performance. The degree of alignment, ψ , is the metric used to indicate the alignment status of the robots. A minimum value of $\psi = 0$ indicates the robots are non-aligned, while a maximum value of $\psi = 1$ means they are perfectly aligned.

Figure 4.2 shows two robots in the swarm and the resultant force \vec{F}_i acting on one of the robots, determined from interactions with all neighbors. The figure indicates that the resultant force has a component perpendicular to the robot's current direction, causing rotation and eventually leading to alignment between the robots.

Although AES-based algorithms show promising results, virtual elastic interactions can cause fluctuations, leading to instability and reduced robustness, especially with noisy measurements. However, viscoelastic interactions can mitigate these fluctuations, improving both stability and robustness. Thus, using viscoelastic links is expected to enhance flock motion while maintaining formation stability. In addition to the model's structure describing collective behavior, parameters play a crucial role in the swarm performance. These parameters are typically tuned empirically, but optimization techniques can significantly enhance performance. Therefore, several works have focused on optimizing AES model parameters to improve the performance of the computer model and make it more suitable for real-world applications using mobile robots. In the work of Raufi et al. [305], a

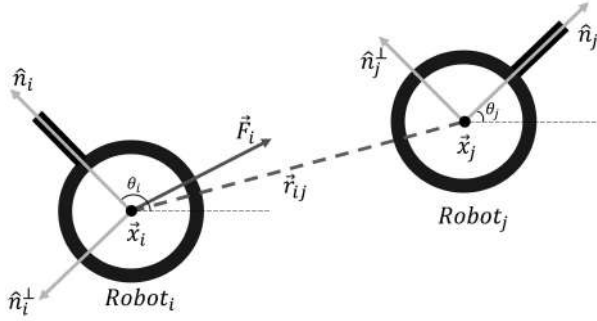


Figure 4.2: The schematic description of elastic interaction (from [26]). ↻

Table 4.1: The values of the control parameters for AES, AES-TCACS, AES-GA and OCM algorithms. ↻

Methods	Model Parameters				
	α	β	k	c	v_0
AES [116]	0.01	0.12	5	0	0.002
AES-TCACS [305]	0.066	0.97	1.28	0	0.05
AES-GA [27]	0.3696	0.9124	3.5329	0	0.05
OCM [28]	0.0262	0.5627	1.0	1.7503	0.075

Tabu Continuous Ant Colony System (TCACS) [193] was used to tune the AES model parameters by minimizing the force among robots and maximizing their alignment. The results showed better performance compared to the original parameters. Other studies [26, 27] improved the collective motion of the AES model using Particle Swarm Optimization (PSO) and Genetic algorithms, respectively. The cost function minimized virtual forces between swarm individuals, alignment error, and convergence time. This approach significantly enhanced collective motion behavior and swarm shape stability, outperforming both the original study [116] and the TCACS-optimized study [305]. However, these works did not consider measurement noise and its impact on collective motion behavior. In a recent study, Bahaidarah et al. [28] proposed an Optimized Collective Motion algorithm that employs viscoelastic interactions between robots to enhance robustness against various disturbances such as measurement noise, environmental factors, and modeling uncertainties. This underscores the algorithm's suitability for real-world robotic applications. The model parameters are automatically tuned using PSO optimization to achieve (i) minimal control effort, (ii) rapid alignment, and (iii) noise robustness.

Figure 4.3 illustrates four different experiments with swarms controlled by different algorithms listed in Table 4.1.

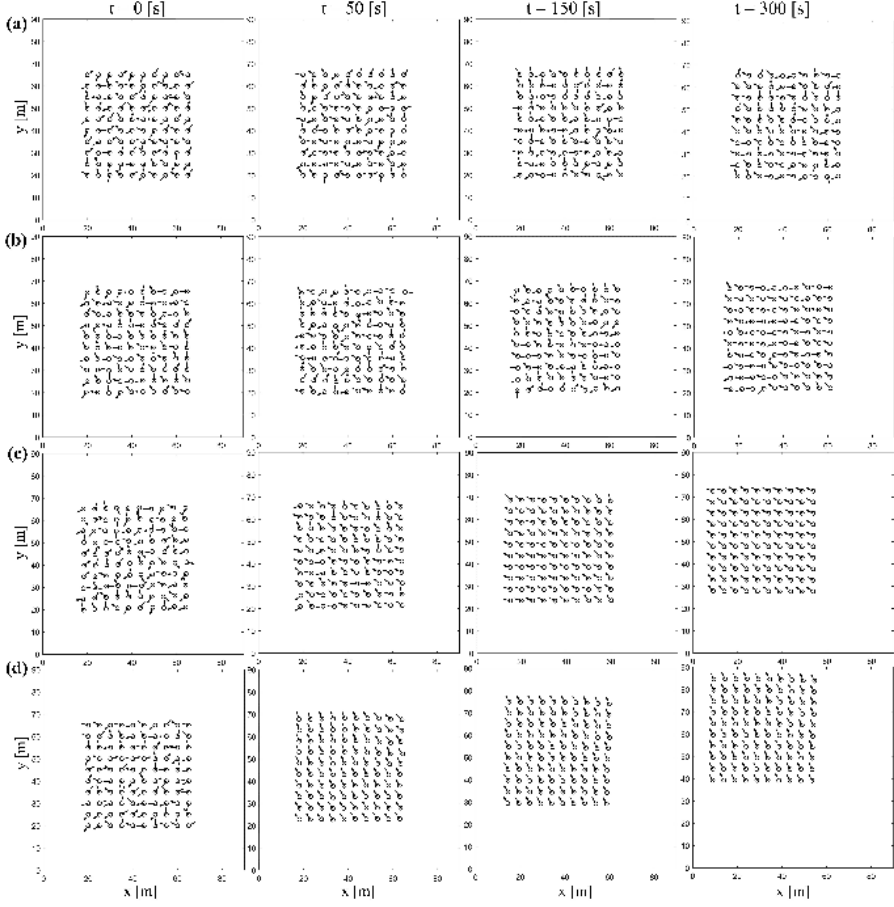


Figure 4.3: Swarm collective exploration controlled by (a) AES, (b) AES-TCACS, (c) AES-GA, and (d) OCM. \square

4.3 Use Case: Strategies to Mislead UAV Swarms

The core concept involves deploying defender UAVs into an attacking UAV swarm with the goal of misleading the swarm without causing further harm. Rather than relying on physical methods to remove the attacking UAVs from the sky or deploying defensive UAV swarms that engage in combat and risk collateral damage, we explore a strategy where a small number of defender UAVs infiltrate the attacking swarm to divert it from its intended mission. Figure 4.4 illustrates this scenario: (a) Shows a swarm targeting a specific objective, while (b) depicts defender UAVs infiltrating the swarm to mislead it. This approach has been largely overlooked in existing research [354], making

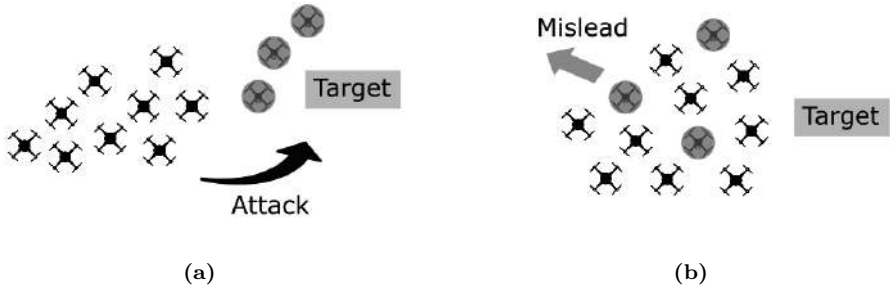


Figure 4.4: Main idea of the scenario, in which (a) UAVs are attacking a target and (b) defenders are induced into the attacking swarm to mislead it [354]. ◻

this work pioneering towards developing an intelligent and adaptable solution for redirecting swarm behavior intended for malicious activities.

In our initial exploration of how to divert a swarm from its mission, we equip both the attacking UAV swarm and the defending UAVs with the same algorithms. Our approach involves first *selecting* appropriate attacking algorithms and then *designing* effective defending algorithms. Given that the hostile swarm is assumed to be targeting a specific objective, we focused on target-oriented swarm algorithms inspired by nature, such as group hunting and foraging behaviors observed in animals and fungi. We selected the Grey Wolf Optimizer (GWO) [247] and the Slime Mould Algorithm (SMA) [228] for this study, both of which emulate natural strategies for locating food sources.

We equip the defending UAVs with the same algorithms as the attacking UAVs, slightly modified to divert the attackers from their original target. The primary goal of the defenders is to steer the attacking UAVs away from their target by influencing their movement. The scenario is set in a continuous 2D area containing a static target M at position $m = [x_m, y_m]$. Here, a swarm of attacking UAVs $a_i \in A$ (where $i = 1, \dots, I$) and a smaller set of defending UAVs $d_j \in D$ (where $j = 1, \dots, J$ with $J < I$) operate according to the local rules of their swarm algorithms. Each UAV is aware of its position, with $p_i = [x_i, y_i]$ for attackers and $p_j = [x_j, y_j]$ for defenders. The attacking swarm aims to locate and reach the target M using the Grey Wolf Optimizer (GWO) [247] and the Slime Mould Algorithm (SMA) [228]. All UAVs are equipped with processing, storage, and communication capabilities.

The defenders aim to redirect the attackers by modifying the fitness function q_m slightly from the original GWO and SMA functions. The modified fitness function also aims to minimize the distance to the target, with a smaller distance indicating better fitness. Fitness values are shared among all UAVs, allowing defenders to inject misleading fitness values to deceive the attacking swarm. The defenders must avoid injecting random values, as this would result in being disregarded by the attackers. Instead, they must move in conjunction with the attacking swarm, gradually diverting it from the target. The fitness

function for the defenders is modeled as follows:

$$q_m(p_j, m) = q(p_j, m) - \phi, \quad \phi \in \mathbb{R}, \quad (4.3)$$

where

$$q(p_j, m) = \sum_j^{\dim p_j} (p_j - m)^2, \quad (4.4)$$

with p_j being the position of the defender UAV, m the position of the target, and ϕ a fixed value allowing defenders to achieve a better fitness relative to the attackers. The value of ϕ is arbitrary and determined through experimentation, where a low value results in minor distraction and a high value in significant distraction.

To effectively steer the attacking swarm, defenders need to move strategically. Preliminary experiments revealed that moving defenders to a fixed point, such as a static fake target, is ineffective due to the dynamic nature of the attacking swarm. Therefore, the next position $p_j(t+1)$ of a defender UAV is calculated as follows:

$$p_j(t+1) = p_j(t) + v_{\max}(t) \cdot (1-c) \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad (4.5)$$

where $p_j(t)$ is the current position, v_{\max} is the maximum velocity of a UAV per time step, and c is the crowd factor, a value in the interval $[0, 1]$. The crowd factor balances between leaving the attacking swarm ($c = 0$) and following it ($c = 1$), allowing defenders to stay close enough to influence the attackers gradually. The vector $[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]$ is of unit length but can be oriented to direct UAVs towards a specific direction (e.g., bottom right).

Figure 4.5 presents the results of the GWO simulations, illustrating (a) the probability h_P of attackers winning as the number of defenders increases, and (b) the discrete probability density of the achieved hits over time with varying numbers of defenders. As anticipated, the presence of more defenders improves h_P , as a greater number of defending fake-wolves exerts a stronger influence on the entire swarm. Details of the results can be found in a paper by Simonjan et al. [354].

4.4 Use Case: The Principle of Hormones for Production Plants

As already described in Section 3.2, the production of logic and power integrated circuits (ICs) in the semiconductor industry is inherently dynamic and complex [138]. Unlike the high-volume production of memory ICs, the logic and power sector features wafer production with a vast product mix, frequent system changes, and a large number of processing steps involving numerous machines [203]. Weekly operations can involve approximately

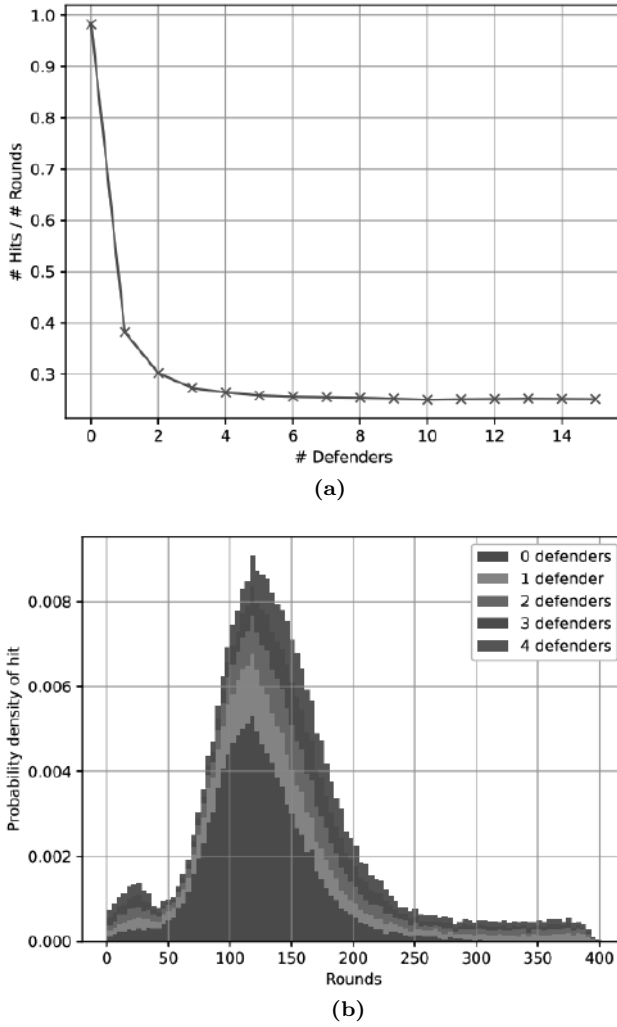


Figure 4.5: GWO simulation results: (a) Probability h_P that attackers win over an increasing number of defenders and (b) probability density of achieved hits over time and increasing number of defenders [354]. ↵

10^5 tasks across 10^3 machines [379]. Optimizing this process for work in progress (WIP) and flow factor presents an NP-hard challenge [130]. Existing dispatching rules and linear optimization methods struggle to handle the NP-hard search space [222], limiting their ability to account for the entire system behavior due to computational complexity, and thus failing to fully exploit optimization potential in job-shop scheduling within semiconductor production systems [203].

To address these challenges, we propose modeling the production plant as a self-organizing system of agents that interact non-linearly, as described in Section 3.2. This approach aims to achieve near-optimal solutions within feasible computation times. Such a system can adapt to environmental changes, scale with the number of agents, and maintain robustness against single points of failure due to its reliance on local interactions [160]. Utilizing local rules and interactions helps circumvent the extensive computation times associated with centrally performed linear optimization.

One promising self-organizing approach involves algorithms inspired by biological hormone systems. These artificial hormone systems mimic the biological endocrine system, which regulates cellular metabolism in the body [358, 387]. As part of the broader class of self-organizing systems, they exhibit properties like scalability, adaptability, and robustness [292], making them suitable for coordinating complex agents in networked technical applications [43, 98, 356]. Such algorithms are particularly valuable for large cyber-physical systems, such as those found in the semiconductor industry, where traditional control or scheduling mechanisms reach their operational limits.

The artificial hormone algorithm is designed using a bottom-up approach to represent both the urgency of processing a lot and the demand for incoming lots by machines. In this system, artificial hormones are generated at machines and diffuse through the production environment, aligning with the processing steps of the lots. The lots function as swarm members that are attracted to machines based on the hormone levels present. The algorithm relies on five key parameters, each with specific mechanisms described below. Table 4.2 provides an initial set of parameter values used for preliminary simulations. The algorithm outlines the calculations and decision-making processes that influence the handling of lots, offering flexibility in implementation. For instance, hormone-related computations can be performed directly at the machines or within a networked monitoring and control layer [101].

Hormone Model Each processing step is associated with a specific hormone type. Hormones can exist at any machine, and multiple hormone types can coexist at a single machine. At each simulation tick, hormones degrade exponentially at a rate α :

$$hormone_amount = hormone_amount \cdot (1 - \alpha) \quad (4.6)$$

An evaporation rate of 0 means that hormones do not degrade, while the maximum value of 1 indicates immediate degradation.

Hormone Production by Machines to Attract Lots Each machine produces a hormone corresponding to its process type. Machines performing the same process type produce the same hormone. If a machine handles multiple process types, it generates hormones for each type in equal parts.

Table 4.2: Suggested algorithm parameters. ↵

Parameter	Value
α	0.3
β	1
γ	0.5
δ	0.2
ε	0.8

Machines aim to maximize their working time by attracting enough lots into their queues. The hormone output of a machine is calculated as follows:

$$hormone_output = \frac{1}{lots_in_queue + \beta}, \quad (4.7)$$

where β is a smoothing factor greater than zero.

Machine Linking A machine A is upstream-linked to machine B if a recipe includes processes of both machines in consecutive steps. If each machine performs only one process, the link strength from this recipe is 1. Otherwise, the link strength is 2 divided by the number of processes supported. If the sequence appears in other recipes, the link strength accumulates.

Hormone Diffusion Upstream When upstream links exist, a fraction γ of the hormone at a machine diffuses upstream:

$$upstream_hormone = hormone_amount \cdot \gamma \quad (4.8)$$

$$hormone_amount = hormone_amount - upstream_hormone \quad (4.9)$$

Each upstream-connected machine receives a proportional share of the upstream hormone

$$added_hormone = upstream_hormone \cdot \frac{link_strength}{\sum link_strengths}, \quad (4.10)$$

where $link_strength$ refers to the upstream link strength for the given hormone between the respective machines, and $\sum link_strengths$ denotes the sum of all upstream link strengths for the hormone from the sending machine.

Hormone Diffusion by Incoming Lots Incoming lots cause a portion δ of the corresponding hormone at a machine to diffuse upstream:

$$upstream_hormone = hormone_amount \cdot \delta \quad (4.11)$$

$$hormone_amount = hormone_amount - upstream_hormone \quad (4.12)$$

The upstream hormone is added to the machine from which the lot arrived, allowing the flow of lots to self-stabilize:

$$added_hormone = upstream_hormone \quad (4.13)$$

Lot Prioritization by Timing A lot's base priority is determined by its remaining Raw Process Time (RPT) and Planned Cycle Time (PCT):

$$base_priority = \frac{remaining_RPT}{remaining_PCT} \quad (4.14)$$

Hormone Attraction of Lots The priority of a lot is influenced by hormones present at the machine where the lot is waiting:

$$attraction = \sum_{i=0} h_i \cdot \varepsilon^i, \quad (4.15)$$

where h_i represents the hormone of the process that is i steps ahead in the lot's recipe, with h_0 being the hormone of the current process. The factor ε indicates the influence strength of the hormone.

The lot's priority is then calculated as:

$$priority = base_priority \cdot attraction. \quad (4.16)$$

At each machine, lots are processed based on their priority, and if batch processing is used, all eligible lots are processed together in a batch.

The algorithm was implemented and tested using a NetLogo simulation model. The simulation results show that the artificial hormone system improves overall production time and the flow factor by approximately 5%. Further details and the evaluation of the algorithm can be found in Elmenreich et al. [101].

4.5 Design Behaviors using the Concept of Evolution

When designing a swarm robotic system, several decisions must be made, including the choice of the robot model and its capabilities, the planned number of robots, the control system, the software model, and the algorithms to be employed. A major challenge is given by the fact that decisions on one aspect can influence the possible degrees of freedom at another level. For instance, choosing a simple robot with a limited number of sensors would necessitate a control system that relies on the close cooperation of a large number of robots to accomplish the task. The challenge of determining the system architecture at multiple levels is akin to the hardware-software co-design approach used for embedded systems [407].

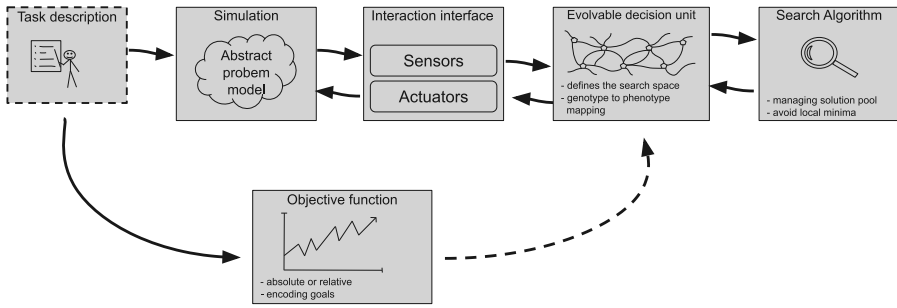


Figure 4.6: Proposed design methodology [112]. ↵

Following a design process supported by evolutionary optimization, we propose a methodology that addresses the key decision points faced by designers of robot swarm experiments. For a given experiment, we identify five major components to be designed, as depicted in Figure 4.6. These five major components are derived from the system architecture of a self-organizing system design process described in Fehervari and Elmenreich [112]:

Task description: Set of requirements that the solution has to meet.

Simulation setup: Describes the simulation model and the relations between the system’s components and their environment.

Evolvable decision unit: The evolvable representation of the local rules.

Interaction interface: Describes how the decision unit interacts with its environment.

Search algorithm: A meta-heuristic algorithm responsible for finding novel and better solutions.

Objective function: The cost function guiding the search algorithm.

Engineering problems usually begin with a task description clearly outlining the goals and constraints. This description serves as a contract outlining the expectations of the desired system at a high level of abstraction. Thus, the task description significantly influences the simulation model used and the target function. In a further step, an efficient system model is developed based on the task description. This model should accurately capture the essential aspects of the system to be simulated while abstracting from unnecessary factors. It is important to note, however, that this step should not delve into how individual components are rendered or how they interact, as that would preempt some important decisions. For this reason, we divide the process into two units: The interaction interface unit and the

evolvable decision unit. The former focuses on strategically planning the interactions between system components and their environment, specifying communication methods (such as sensors) and the underlying interfaces (such as protocols). The latter focuses on the actual representation of the system components, which may include one or more model types depending on the homogeneity of the system. The reason for separating this part from the system model is the need for an evolutionary approach, which requires “evolvable” representations. In other words, components must be able to generate adaptive genetic diversity, which is typically achieved by defining mutation or crossover operators [10]. Given the variety of available representations, careful consideration by the designer is essential because of the often different properties associated with each representation.

An important part of the evolutionary process is the algorithm that optimizes candidate solutions. This can be viewed as an efficient search algorithm that operates on candidate solutions represented by the component representation. Both the component representation and the search algorithm can be implemented independently against a well-defined interface defining how the search algorithm can access and evaluate candidate solutions. This allows optimization and candidate representation to be chosen separately. There can be an advantage of using a particular search algorithm with a particular candidate representation; for example, evolving neural network structures would benefit from a niche formation mechanism that prevents innovations from dying out prematurely, and restricting recombination to compatible descendants among the pool of neural network solutions [363].

To guide the evolutionary process towards a desired solution, an objective (or fitness) function needs to be defined that either indicates the quality of a particular candidate solution or at least allows for the comparison of the quality of two candidate solutions. Designing a proper objective function is crucial for the overall evolutionary optimization approach. The objective function should be readily accessible, either continuous or sufficiently fine-grained, and should clearly favor the desired solutions while avoiding the reward of undesirable ones. A good objective function should not only assess high scores to proper solutions but also form a smooth fitness landscape, avoiding, whenever possible, local maxima that could trap the evolutionary search. Designing such a practical function is often difficult, even for domain experts [183].

4.6 Framework for Evolutionary Design

4.6.1 Architecture

The architecture of FREVO is strictly component-based, where the steps of evolutionary design are divided into individual components. This architecture makes it possible to develop a single component and easily replace individual components. Therefore, different configurations can be easily evaluated to find the most appropriate configuration for a given problem. Each component

implements a particular feature of the evolutionary approach. The *problem* component defines the specifics of a CPS controller, environment, and fitness function. The *representation* component defines how the CPS controller is represented. The *optimization* component defines the method for finding the optimal candidate representation. The *ranking* component defines how the candidate representations are ranked based on their performance.

FREVO is implemented in the Java programming language and follows the object-oriented programming paradigm for encapsulating components. Each component is defined by an abstract class, standardizing the interfaces between parts such as for example agent representation, simulation and optimization algorithm. Therefore, new components can be used in a setting with existing components completing the overall system. Creating a component is guided by a built-in component generator, which helps the software developer generate the required code framework in the context of the class hierarchy. FREVO is released under an open source license, supporting the sharing of research ideas and technical solutions. The source code is freely available.¹

The FREVO GUI guides a user step by step through the configuration process. This is done by selecting a component for each task of the evolution process. A single configuration of components is called a FREVO session. Such sessions and the results of the optimization process can be exported and saved for import for later use.

Problem Definition

The problem definition in FREVO defines the parts related to the task to be achieved, including the environment and the interaction possibilities of the agents, but not the implementation or optimization of the behaviors. The problem definition thus consists of the formulation of a goal, the simulation model, and the interaction interface for the agents within the simulation. In the case of CPS systems, the interaction interface models the behavior of sensors and actuators, and the environment is implemented as a physical simulation model. Candidate controller representations are linked to sensors and actuators, with each sensor serving as input and each actuator as output.

The problem definition also implements an objective function and the means for evaluating the values of the objective function in the course of one or multiple simulation runs. Thus, the fitness assessment typically occurs in the phenotype space, where the behavior in the simulated environment is observed. The genotype, which refers to the implementation of a controller, typically cannot be assessed without executing it in a simulation. The objective function then guides the heuristic optimization process in finding the optimal candidate representation.

There are two problem definition types: *AbstractSingleProblem* evolves a CPS candidate for cooperative task execution in a swarm setup,

¹FREVO Github: <https://github.com/smartgrids-aau/Frevo> [Online; accessed: 05-February-2025].

while *AbstractMultiProblem* involves evolving and evaluating multiple representations against each other, suitable for competitive multi-agent systems. The former applies to homogeneous multi-agent systems, ranking candidates by fitness, while the latter suits competitive systems, with fitness evaluated relative to other agents. Tournament algorithms rank candidate pools, as seen in scenarios like soccer games: To determine the best team among several soccer teams, the teams play each other until a tournament winner is determined. An example of a soccer team use case, where two teams compete against each other in an evaluation, is presented in [111].

Developing a new problem definition involves defining the interface between sensor inputs, actuator outputs, and candidate representation; implementing simulation for evaluation; and calculating fitness based on performance measures, all guided by predefined system interfaces. Given existing components, developers can focus on implementing new problems without concern for representation, optimization, or ranking components. Complex simulations can leverage external simulators from model libraries or custom interfaces as detailed in Section 4.6.4.

Candidate Representation

The candidate representation models the internal structure of the CPS controller. It is a generic structure that is evolvable, for instance, an Artificial Neural Network (ANN). It encodes the behavior of the CPSs, including reactive behavior to stimuli via the sensors. A candidate representation, when viewed together with the problem definition, represents a possible solution to the given problem. Every representation must define the genetic operators such as mutation, crossover and selection. For supporting the user in analyzing the representation, different output formats can be implemented. Typically, the representation is derived from the *AbstractRepresentation* class and is common among all agents. For heterogeneous multi-agent problems, one can choose the bulk representation to evaluate a set of candidates with distinct representations.

Currently, FREVO supports the following representations:

- *Fully-meshed net*: This is a recurrent ANN with a fully meshed architecture providing mutual connections between any two neurons. Throughout evolution, both the biases of the neurons and the connection weights undergo alterations. During run-time, this network has some memory due to possibilities of feedback loops.
- *Three-layered net*: This is a feed-forward, non-recurrent ANN incorporating one hidden layer. The biases of the neurons and the connection weights evolve similarly. Unlike the fully meshed ANN, this architecture is tailored for simpler problems, substantially reducing the search space.

- *NEAT*: In this model, the ANN adapts by evolving the connectivity between neurons, following the NeuroEvolution of Augmenting Topologies (NEAT) method introduced in [363].
- *HebNet*: Incorporating Hebbian learning, this recurrent, fully interconnected ANN endows synapses with plasticity enabling real-time learning. In this setup, both plasticity and initial weights are evolved with Hebbian learning occurring during the simulation runs.
- *Simple bulk representation*: This is a combination of multiple representations as described above.

Optimization Method

The optimization method aims to find the candidate representation that maximizes fitness, as the problem definition specifies. It employs genetic operators from the candidate representation to generate new candidates in each generation, replacing the least effective candidates. This iterative heuristic search progressively produces candidates with improved performance. The search continues until a termination criterion specified within the optimization method is met. Examples of termination criteria include reaching a maximum number of generations or consecutive generations without fitness improvement.

The optimization methods currently offered by FREVO are as follows:

- *Random search*: A baseline comparison technique in which randomly generated candidates replace those with low fitness.
- *NNGA*: This is an Evolutionary Algorithm (EA) that maximizes the population diversity. It supports multiple populations and several ranking algorithms. It is based on the Neural Network Genetic Algorithm (NNGA) described in [100]. It is well-suited for evolving any type of representation.
- *GASpecies*: This is an EA that classifies candidates into species. Species are determined by a similarity function defined in the candidate representation. Within each species, candidates share the same fitness value.
- *CEA2D*: It is a cellular EA that arranges all candidates on a 2D torus surface. Genetic operations are executed in a local context, resulting in improved diversity and slower convergence compared to traditional evolutionary algorithms.
- *Novelty search*: This is an EA that prioritizes behavioral diversity over fitness. Its implementation is based on rtNEAT in [362].
- *Novelty species*: This is an EA that rewards behavioral diversity across different species.

Ranking Algorithm

The ranking algorithm sorts candidate representations based on their performance, i.e., fitness value. The ranking algorithm is also responsible for parallelization to decrease the overall simulation time of the optimization process by deciding which evaluations can run in parallel on a multi-threaded system. Problems where the fitness of a candidate can be obtained directly by a simulation run require a so-called *absolute ranking*. Two types of absolute rankings are currently implemented in FREVO:

- *Absolute ranking*: A ranking algorithm that sorts candidates by the fitness value returned from the problem component. It supports multi-threading to decrease the time needed for optimization.
- *Novelty ranking*: A ranking algorithm that sorts candidates based on their novelty in the behavioral space.

In contrast, problems that necessitate pairwise evaluations between the candidates follow the structure of an *AbstractMultiProblem*. For this type, a full tournament ranking and a ranking based on the Swiss system are provided. The Swiss system, inspired from chess tournaments, can provide a ranking with fewer comparisons at the cost of ranking accuracy [99].

4.6.2 Graphical User Interface

The GUI of FREVO provides a convenient means to assess an evolutionary design process component swiftly, showcasing the underlying modular architecture (see Figure 4.7). This visual representation illustrates FREVO's GUI, where an example problem has been evolved, with the “Select Problem Component” window displayed.

The top-left “Configure Session” panel facilitates session configuration by guiding users through selecting a problem, optimization algorithm, candidate representation, and ranking method step by step. This streamlined process expedites the testing of new components. Each selection prompts a new window for configuration, with options for fine-tuning parameters.

Below, the “Control” panel enables users to initiate, halt, or reset the optimization process. Progress can be tracked in real-time through the “Statistics” panel on the right and the “Console” panel below. The “Statistics” panel presents graphs illustrating the evolution of fitness and diversity across generations, while the “Console” panel displays output from active components. Upon completion of optimization, results can be accessed from the panel on the left that let's the user inspect the latest generation. Here, users can save or replay the results for further analysis. Clicking “Replay” allows a closer examination of each candidate representation from the last generation, including visualization of resulting behavior in simulation.

The top menu provides options for saving and restoring previous sessions and managing components, including the “Component Creator”.

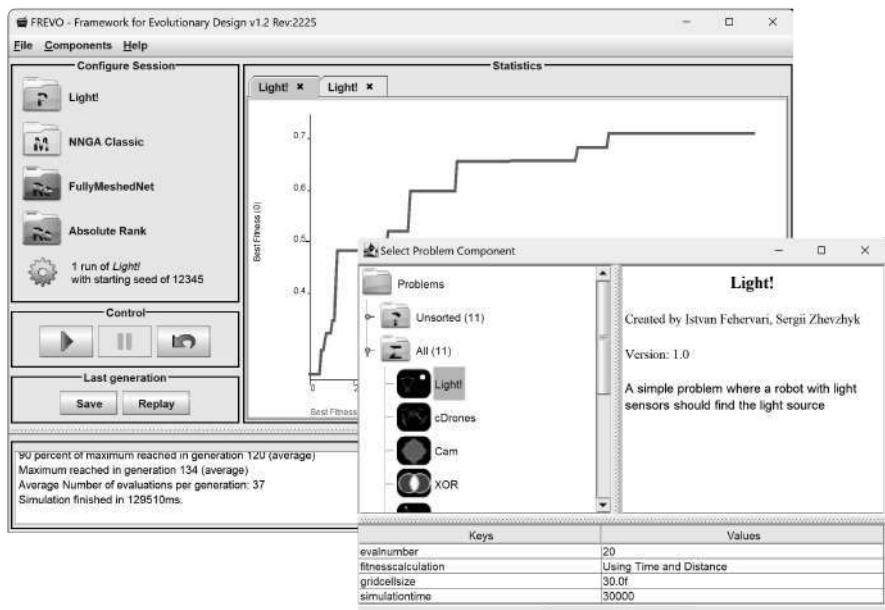


Figure 4.7: Screenshot of the FREVO GUI showing the evolution of an example problem. ↵

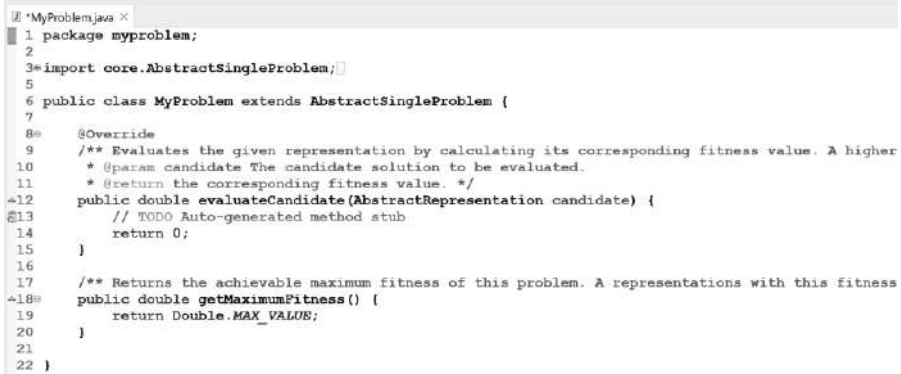
This tool automates skeleton code generation, aiding software developers in implementing new components for the model library.

4.6.3 Workflow

When executing FREVO with existing components, initiating optimization requires just a few clicks. First, users select the desired problem, followed by the optimization method, candidate representation, and ranking algorithm. Optimization continues after clicking the play button until the selected termination criterion is met. The statistics panel displays two graphs: One depicting the best fitness value of each generation and another showing the diversity in each generation (refer to Figure 4.7).

The component creator in the top menu can be utilized to model a new problem. A code skeleton is generated after selecting the component type, name, package, and description. This code is placed in a subdirectory of the components directory in FREVO, accompanied by comments guiding further implementation (see Figure 4.8). Additionally, an XML file named after the component is created, allowing the definition of configuration parameters, sensor inputs, and actuator outputs for the CPS.

The primary task involves implementing the *evaluateCandidate* method, where the candidate representation is evaluated through simulation, either



```

1 package myproblem;
2
3 import core.AbstractSingleProblem;
4
5
6 public class MyProblem extends AbstractSingleProblem {
7
8     @Override
9     /** Evaluates the given representation by calculating its corresponding fitness value. A higher
10      * @param candidate The candidate solution to be evaluated.
11      * @return the corresponding fitness value. */
12     public double evaluateCandidate(AbstractRepresentation candidate) {
13         // TODO Auto-generated method stub
14         return 0;
15     }
16
17     /** Returns the achievable maximum fitness of this problem. A representations with this fitness
18     public double getMaximumFitness() {
19         return Double.MAX_VALUE;
20     }
21
22 }

```

Figure 4.8: Screenshot of the source code skeleton of a newly created problem. ↵

directly within FREVO or by invoking an external simulator. Consequently, the environment and CPS must be implemented. Sensor inputs of the CPS are passed to the *getOutput* method of the candidate representation, which returns the actuator output(s). A suitable performance measure is then implemented to compute the fitness value of the simulation run.

Once a component's implementation is complete, it needs to be compiled and automatically loaded upon FREVO launch. The new component then appears in the component selection window, enabling the aforementioned workflow. This process remains consistent when creating other components, with the generated code skeleton and implementation tailored to their specific requirements.

4.6.4 External Simulators

External simulators are used in domain-specific simulations. To achieve this, the *evaluateCandidate* function of the problem component must invoke the simulator and provide the candidate representation (i.e., the CPS controller). Consequently, the simulator must incorporate the exact representation to enable FREVO to evolve an optimal solution. This can be accomplished through two approaches: In a compact way through code generation that is then run in the simulation, or in a modular fashion, by transmitting only the inputs and outputs between FREVO and the simulation.

In the compact approach, the simulator's code needs to be recompiled with the newly created representation in each generation. Subsequently, the simulator is executed directly from FREVO, returning the fitness value for optimization. This method is preferable for CPSs lacking file system or network communication capabilities.

Alternatively, the modular approach entails implementing the simulation after achieving the desired candidate representation. In each generation, the parameters defining the representation are then transmitted to the simulator, either through files or network communication. The actions performed by

individual CPSs within the simulator are fully determined by the candidate representation, whose parameters are evolved by FREVO. FREVO executes the simulator, which logs the performance measure into a log file. Upon completion of the simulation, the corresponding fitness value is computed by the *evaluateCandidate* method.

4.7 Why Evolutionary Optimization Needs Simulation

In this chapter, we have explored the design of self-organizing systems through evolutionary methods, focusing specifically on swarm robotics. We introduce the FREVO framework, which leverages evolutionary optimization to address challenges in decision-making and hardware-software co-design. FREVO is a modular component-based system that can be adapted for various applications. Its components are categorized to facilitate simulation, representation of candidates, optimization methods, and algorithm ranking. The system includes a user-friendly GUI for configuring sessions, selecting components, and executing the optimization processes.

Typically, evolutionary approaches rely on assessing the value of a fitness or cost function for a given implementation. This often involves conducting experiments where swarm members interact within an environment similar to their intended operational context, collaborating to solve typical problems. However, fitness values typically are obtained by post-experiment analysis, which in turn demands reinforcement learning techniques capable of handling delayed rewards. While theoretically feasible with real hardware, the practical constraints of conducting thousands of experiments can be prohibitively cumbersome. The challenges include not only the logistical issues of experimentation but also the substantial costs associated with deploying a large number of capable swarm robots, especially when such robots are not mass-produced. Consequently, simulation becomes an indispensable tool in swarm engineering.

In the upcoming chapter, we will explore the pivotal role of simulation in the development and analysis of swarm robotics and cyber-physical systems. We will examine why simulation is an essential tool for advancing our understanding of swarm dynamics and overcoming the practical challenges associated with real-world testing, which can be complex, costly, and logistically demanding. The chapter will provide a comprehensive overview of different swarm simulation techniques, including agent-based models and multi-agent systems. We will review a range of simulation tools and platforms, focusing on their capabilities, advantages, and limitations. Emphasis will be placed on the practical aspects of using these tools, such as cost efficiency and the ability to handle large-scale swarm scenarios in a time-efficient manner, making them especially valuable for researchers with limited resources. We will also address the inherent challenges of swarm simulation, including

the limitations in modeling accuracy and the difficulties in replicating real-world conditions. The discussion will cover how simulation integrates with hardware development to validate results and how emerging technologies, like AI and machine learning, are transforming the landscape of swarm simulation. Through this exploration, we aim to shed light on how simulation can effectively complement physical experiments, offering insights and solutions for designing, testing, and optimizing swarm systems. We will highlight both the benefits and drawbacks of simulation platforms, drawing on examples such as the bio-inspired swarm aggregation mechanism BEECLUST and its various simulations across different frameworks. By understanding these dynamics, researchers can better navigate the complexities of swarm robotics and cyber-physical systems, ultimately advancing the field with more innovative and cost-effective approaches.

Chapter 5

Simulating Swarms of Cyber-Physical Systems

Real-world implementation and evaluation of a swarm robotics scenario that often requires a huge number of robots is very complex, costly, and requires large settings. Therefore, simulation platforms for swarm systems are a very popular and efficient choice for a research project in an early stage. This motivated researchers to develop many swarm simulation platforms that successfully replicated the bio-inspired mechanisms of swarm robotics. These simulation frameworks, predominantly open-source, have enabled several research groups, even those with limited budgets, to conduct swarm studies involving a large number of robots.

There are many benefits and drawbacks in using simulation software instead of a real-world robotic platform. The main advantage is the cost efficiency of the simulation platforms that are mostly designed for open-source operating systems (OS) such as Linux. Although their speed and performance are limited to the host system performance, they can implement complex swarm tasks in a low-cost, time-efficient manner. Additionally, simulation software offers various benefits that make them an ideal choice for early career researchers and students, such as the ability to share code and experimental configurations with other researchers. This process allows to easily maximize the replicability of research works. The biggest drawback in using simulation software is the poor modeling and reduced abstraction that introduces inaccuracy when simulating large numbers of robots in a swarm scenario. Even internal and external conditions are hard to model and often present a poor environment for virtual swarm robotics platforms. Additionally, there are many uncertainties and complexities at various levels, from individual behavior to collective behavior. Consequently, the results obtained from simulation software often do not align with the outcomes observed in real-world-robot experiments.

Table 5.1: List of simulation platforms commonly used for swarm robotics. ۞

Simulation Platform	Operating System (OS)	Level	Open-source	Application	2D/3D
AirSim [348]	Linux/Win	Physics	Yes	Generic	3D
ARGoS [290]	Linux/Mac	Physics	Yes	Swarm	2D & 3D
BeeGround [229]	Linux/Mac/Win	Physics	Yes	Swarm	2D & 3D
Gazebo [209]	Linux/Mac/Win	Physics	Yes	Generic	3D
Kilombo [179]	Linux	Abstract	Yes	Swarm	2D
MASON [236]	Linux/Mac/Win	Abstract	Yes	Generic	2D & 3D
MESA [239]	Linux/Win	Abstract	Yes	Generic	2D
NetLogo [382]	Linux/Win	Abstract	Yes	Generic	2D
OpenHRP [190]	Linux/Win	Physics	Yes	Generic	3D
PyCX [324]	Linux/Win	Abstract	Yes	Generic	2D
SCRMAGE [81]	Linux/Mac	Physics	Yes	Generic	3D
Stage [392]	Linux/Mac	Physics	Yes	Swarm	2D
Swarm-sim [60]	Linux/Mac/Win	Abstract	Yes	Swarm	2D & 3D
USARSim [56]	Linux/Mac/Win	Physics	Yes	Generic	2D & 3D
V-rep [315]	Linux/Mac/Win	Physics	No	Generic	3D
Webots [246]	Linux/Mac/Win	Physics	Yes	Generic	3D

As an example for the extensive work on the implementation of a robotic swarm scenario, the bio-inspired swarm aggregation mechanism, BEECLUST [327], has been simulated numerous times with different settings in various simulation frameworks. These setting include pheromone-based aggregation in PyCX simulator [23], a study of swarm interaction in NetLogo software [37], fuzzy-based decision making in Stage [22], aggregation in a complex environment in BeeGround [401], and a source exploration scenario in Webots [11]. However, while these studies implemented the same bio-inspired aggregation behavior, they did not consider the honeybees' diversity (behavioral and physical heterogeneities). Hence, they did not replicate similar results observed from real-life honeybee and real-robot swarm experiments [374, 329] as this abstraction level was already too high. Similar limitations were also seen in other swarm algorithms, such as flocking.

Table 5.1 lists some of the popular softwares that have been created for the usage of simulating CPS swarm scenarios and applications. Simulation softwares can model agents at various complexity levels, ranging from abstract simulators like PyCX and NetLogo that do not represent the physical properties of real robots, to physics-based realistic simulators like ARGoS and Gazebo which consider all the physical parameters of a mobile robot. The reality level of the simulator directly relies on the PC's processing power. Nowadays, high-performance PCs allow us to implement large swarm experiments with physics-based simulation very close to reality.

5.1 Simulation Requirements

The main reason for using simulation software instead of real robots is to have a cost-effective approach to investigate new ideas in a time-efficient manner. While a trade-off exists between the precision of implementation and associated costs, the consensus among researchers is that in the initial phases

of a project involving swarm scenarios, simulations represent the optimal approach. As a result, simulation software for CPS swarms are becoming increasingly popular among small and large research labs. Therefore, to perform a reliable swarm experiment in simulation, the utilized simulation platform must have the following criteria:

1. It must support a fully **decentralized** process in which each robot can have its own control system and parameters locally. It is very important that each robot in the simulation platform operates in a decentralized manner with its own internal controller. Although having a central controller can simplify the implementation of a swarm experiment, replicating this setup in a real-world scenario would be complex especially when dealing with a large number of robots or when robots are deployed over vast distances such as 100s of kilometers apart.
2. It should consider the **physical properties** of a real-world robotic platform as much as possible, including aspects such as motors and gears, friction, mass, power consumption, and more. Nonetheless, it is crucial to bear in mind that the ultimate objective is to deploy swarm behaviors in real-world scenarios, resolving human problems effectively in the practical application phase. Therefore, a realistic simulation of a swarm system will allow us to replicate the behaviors of the swarm on real robots with minimum effort.
3. It should replicate the physical properties of the real-world **environment**, such as light, shadow, color, and texture. Similarly to the previous case a simulation must be capable of handling environmental conditions and replicating physical properties as closely as possible to real-world conditions.
4. The simulated **sensors** must be realistic enough to model various physical measurements such as velocity, force, position in 2D and 3D, and heading angle. However, the real-world implementations always include some level of false data due to noise in the sensory system. Hence, it is instructive to consider this noise incorporating both systematic and random errors.
5. It should support **long-term autonomy** and provide enough memory and processing power so that a swarm can converge to a stable state. This is also important when we implement the swarm behavior using real robots that need to work hours and hours before recharging their batteries or transferring recorded data to an external storage facility.
6. Last but not the least, the simulator must support **open-source** development so that researchers can customize it to fit their applications. Commercialized simulation software that lacks support for open-source development will not serve as a versatile platform for everyone, from early career researchers to established researchers.

A CPS swarm simulator must follow these essential criteria to enable a reliable swarm experiment and be an attractive simulation platform that maximizes replicability. However, the significance of each criterion depends on the type of simulation output being planned. For example, in using a simulator to act as a 3D space containing particles like robots acting as gas molecules, the physical consideration is limited to molecules' properties rather than defining motors and gears. Therefore, we first need to define whether we go for an abstract or a physics-based simulation software.

5.2 Abstract Simulation

Abstract simulation tools, also known as kinematic-based or particle-based simulations, are powerful tools for the conceptual investigation of swarm behaviors. However, these simulation platforms do not consider the physical properties of the systems; hence, they allocate entire processing power on executing inter-agent interactions in a very large-size swarm. One of the early swarm simulation examples was Boids model developed by Reynolds [311]. The swarm flocking was achieved through three behaviors: I.e., collision avoidance, velocity matching, and flock centering. These behaviors were implemented with the assumption that individual agents have access to the distances and directions of their local neighbors. Other related studies proposed mathematical models describing the collective motion of the swarm, such as the model proposed by Vicsek et al. [394]. In a very interesting work by Schmickl et al. [328], they used an abstract simulation to show a fascinating behavior of a life-like system. They simulated thousands of particles in tens of thousands of simulation runs to show emerging patterns of artificial cell life cycles. In this kind of simulation, each agent is a particle in a 2D or 3D space that freely roams and interacts with others. Parameters such as linear and angular velocities, radius of sensing, size of the agent, basic dynamic model, and more, are yet to be considered in the experiments. In another study, Ferrante et al. [116] introduced a self-propelled collective motion mechanism using a two-dimensional active elastic sheet (AES) model to simulate swarm agents as particles using an abstract simulation. This type of particle-based setting was also used to model interaction networks for collective motion [386]. The AES model was adopted in other studies and several extensions of the model with optimization approaches were implemented using abstract simulations [28, 26].

Abstract simulation is extensively utilized to explore control engineering mechanisms implemented at a theoretical level. These studies are typically conducted using general programming platforms such as MATLAB[®]. For example, in [172], a swarm coordination protocol was investigated considering the non-linearity of the agents' dynamics, which mainly focused on the design of a robust control system for robotic swarms. In another study focused on the robotic shepherding task [171], an abstract simulation of sheep and dogs was developed considering the real dynamics of shepherding dogs to the sheep. The

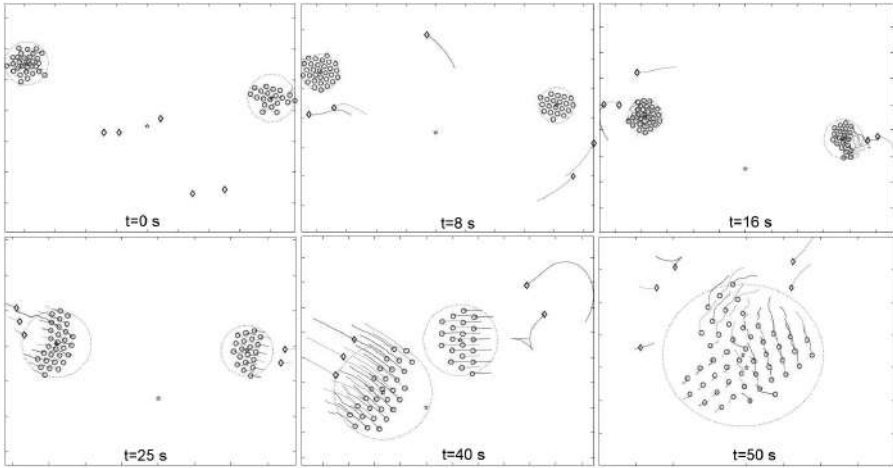


Figure 5.1: Example of a shepherding scenario in an abstract simulation. The positions of the dogs are shown with blue diamonds and the sheep with red circles. The target point is marked by a red star. At the beginning, the dogs were scattered around the target area. From 25 to 40 seconds, the sheep were guided towards the goal. After 40 seconds, one group of sheep had arrived and the dogs began to assist the other flock. At 50 seconds, the two flocks of sheep were combined into one large herd and the herding task was completed. ◀

simulation output offered valuable insights before the actual implementation of the robotic system. Figure 5.1 illustrates the shepherding control strategy and the agents' trajectories in an abstract simulation. Similar implementation of the theoretical swarm system was also utilized in several research works including, formation control [409, 408, 410].

Many studies opt for abstract simulation due to the ease of implementing swarm systems in this environment. In this chapter, we introduce an open-source abstract simulation framework programmed in NetLogo, SwarmFabSim [390].

5.2.1 SwarmFabSim: A NetLogo Implementation

NetLogo is a popular agent-based simulation platform widely utilized for simulation of self-organized systems in research and education [404]. It is a freely available tool that boasts extensive documentation and is actively maintained, ensuring a robust and stable code base. NetLogo provides users with numerous extensions, further expanding its functionality and versatility for various simulation purposes. The simulation software is widely recognized for its extensive use in educational settings, particularly for teaching agent-based modeling and complex systems. However, it is important to note that NetLogo is not limited to educational purposes only. It has demonstrated its maturity as a platform capable of conducting simulations involving several

thousand agents in a reasonable computing time. This has been confirmed through studies such as those by Railsback et al. [303] and [302], which emphasize the platform's ability to handle large-scale simulations effectively. For our own performance results we refer the reader to Umlauf et al. [390]. The NetLogo homepage proudly showcases a vast collection of over 3000 research papers published in the last decade that have employed NetLogo as their simulation platform of choice. This extensive list is a testament to the widespread adoption and credibility of NetLogo within the scientific community for conducting a diverse range of simulations and research studies.

NetLogo provides researchers and modelers with an interactive user interface, enabling them to easily prototype and experiment with their models. Its user-friendly interface allows for intuitive visualization of model behavior and dynamics, facilitating the exploration and analysis of simulation results. Moreover, NetLogo offers a powerful feature called BehaviorSpace, which enables the configuration of batch simulations. Researchers can define multiple parameter settings and specify the desired number of replications to be run. BehaviorSpace automates the simulation runs, collecting and logging the results to files for further analysis. This capability allows efficient exploration of different scenarios, sensitivity analysis, and statistical analysis of simulation results. The log files generated by NetLogo simulations can be easily postprocessed using various tools, including popular statistical analysis software such as R or spreadsheet applications such as Excel. Researchers have the flexibility to choose their preferred tool for statistical evaluation, data visualization, and further analysis.

Furthermore, NetLogo supports direct integration with other programming languages, such as Python. This allows researchers to take advantage of the capabilities of Python libraries and tools for data manipulation, analysis, and visualization. The seamless integration between NetLogo and Python [151] or R [381] allows the use of advanced statistical methods, machine learning algorithms, and custom analysis pipelines to gain deeper insights from simulation results. Overall, the combination of NetLogo's log files and its interoperability with other programming languages provides researchers with a comprehensive toolkit for analyzing and interpreting simulation outputs in a flexible and customizable manner.

In NetLogo, simulations are time-based and operate on a discrete scale using ticks. Agents in the simulation are categorized into different types, called "breeds", and these agents can interact with each other in various ways. These interactions can occur through direct proximity to a two-dimensional plane of patches, through connections defined by a network topology, or indirectly through the use of residual information in the environment.

We implemented the SwarmFabSim simulation framework in NetLogo implementing the use case and the modeling approach for swarms in production plants as described in Section 3.2. This framework supports dispatching and scheduling modes, single-lot oriented and batch machines, and an arbitrary number of machine and lot types. SwarmFabSim is a modular system comprising multiple code modules that communicate through

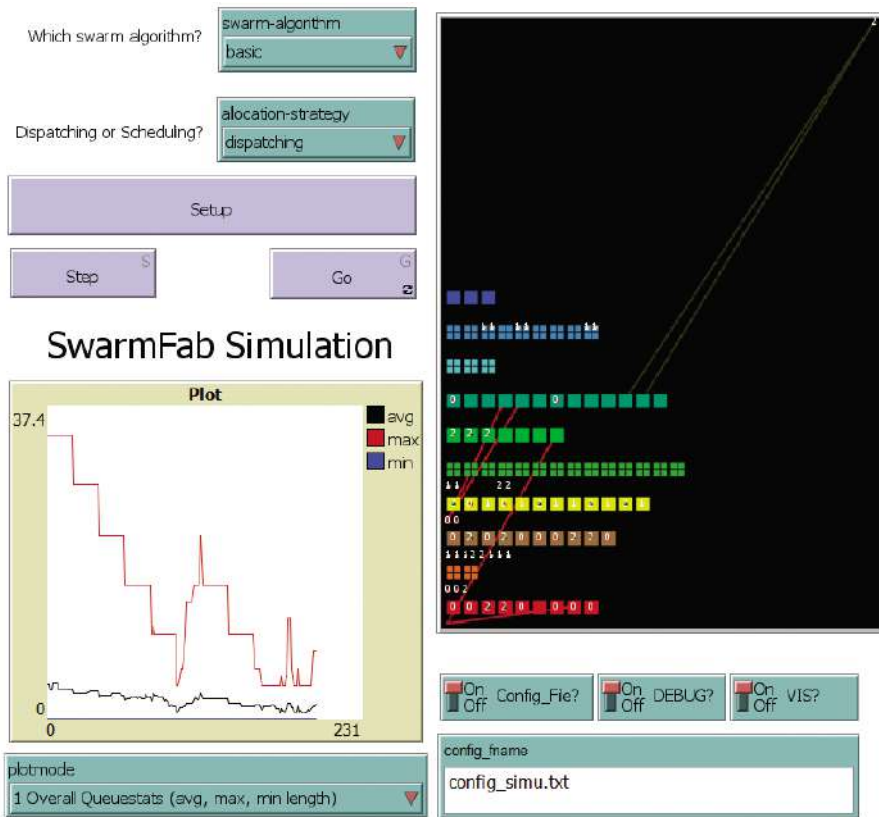


Figure 5.2: SwarmFabSim: Screenshot of the user interface. ↵

a callback architecture. The user interface, as depicted in Figure 5.2, along with the configuration files, enables users to interact and customize the SwarmFabSim framework to align with their specific requirements of the fabrication model.

The simulation scenario is defined using a set of plain text configuration files:

META contains the names of MFILE, RFILE, and LFILE config files. This file bundles the associated config files together. The name of the META config file to be used is set through the “config_fname” input field on the UI or the BehaviorSpace settings.

MFILE is a configuration file that encompasses the machine definitions within the simulation scenario. For each machine type, denoted as m , this file specifies various attributes:

- **Process ID:** It determines the specific process, represented by P^m , that the machine can perform.

- **Number of Machines:** It denotes the total count of machines available for this particular type.
- **Processing Time:** It indicates the time required for the machine to complete a single task.
- **Batch Size:** This parameter defines the number of lots that can be processed as a batch. If the batch size is set to one, the machine operates in a single-lot orientation.
- **Maximum Waiting Time:** For a batch machine, this value indicates the longest time the machine will wait for a batch to be filled before starting the process.

RFILE serves as a repository for all production recipes, denoted as R^t , which are utilized for manufacturing various lot types t . These recipes are essentially straightforward lists that outline the sequential process steps, represented by P^m , in the specific order they need to be executed.

LFILE specifies the production quantities for each lot type t based on their corresponding recipe R^t . It defines the number of lots that should be produced for each specific lot type, indicating the desired quantity of output according to the prescribed recipe.

The callback architecture depicted in Figure 5.3 is implemented through the API defined in the file `hooks.nls`. The main code invokes a hook function contained in `hooks.nls` whenever an algorithm has the potential to perform an action. Subsequently, the corresponding algorithm function is invoked from the hook function, enabling the desired functionality to be executed. In order to install an algorithm, you need to provide a file named “`algorithm-name.nls`” that implements the algorithm based on the hook API. Additionally, you should add the algorithm to the UI chooser labeled as

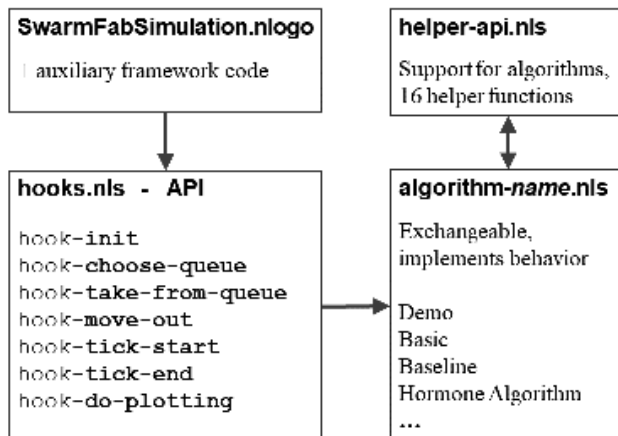


Figure 5.3: SwarmFabSim architecture [390]. ↱

“algorithm” to make it accessible for selection. Furthermore, the algorithm functions should be incorporated into the appropriate hook functions within the `hooks.nls` file for callback purposes. It is also possible for algorithms to utilize helper functions available in the “`helper-api.nls`” file to streamline their implementation and enhance their functionality. These helper functions offer additional functionality and can be leveraged by algorithms to facilitate their operations.

The reader is considered to be aware that while our approach draws inspiration from semiconductor manufacturing, it can be easily adapted to other industries that employ flexible job-shop scheduling. This adaptability is achieved by modifying the configuration files that define the specific parameters of the given industry setting.

The complete NetLogo source code of SwarmFabSim, along with a collection of configuration files, is openly accessible in a dedicated GitHub repository [389]. Users can freely explore, utilize, and contribute to the development of SwarmFabSim through this open-source platform.

5.3 Physics-based Simulation

Physics-based simulation considers (all) the physical parameters of a robot and the surrounding environment. This type of simulation requires higher processing power; hence, a simulation task becomes more complex when the population of the swarm is increased. Many simulation platforms support physics-based simulation, as mentioned in Table 5.1. ARGoS¹ and Webots² are two successful platforms that are widely used for swarm robotics applications. They generate realistic outputs for multi-robot experiments by implementing robot and robot-to-robot physical contact. The main drawback of using the physics-based simulation for large swarm experiments is the computational complexity and limitations resulting from the memory size and processors of the host computer. Although this limits the implementation of large population experiments, there are programming techniques that reduce the issue, e.g., deactivating visualization of experiments. There are many successful implementations of swarm behaviors using physics-based simulation, such as task allocation [48], self-adaptive communication strategy [119], cooperative navigation [95] and swarm controller design [124] in ARGoS; collision-free flocking [29] and formation control [169] in Webots; aggregation [22] and foraging [2] in Stage; and swarm navigation and collision avoidance [264, 309] in Gazebo.

Most simulation software contains a comprehensive list of libraries for commonly used robots; hence, we can easily select a robot from the list and use it. Figure 5.4 illustrates an example of mobile robots available on the Webots platform. Furthermore, most simulation platforms allow us to define a new robot and import the robot’s mechanical design, sensory system, and

¹ARGoS website: <https://www.argos-sim.info/>[Online; accessed: 19-December-2024].

²Cyberbotics website: <https://cyberbotics.com/>[Online; accessed: 19-December-2024].



Figure 5.4: Physics-based models of some of the available robot libraries in Webots software; (left) large and (right) small mobile robots. ↵

physical properties. For example, in [11], a simulated Mona robot model was developed and used to implement an exploration scenario in a swarm system (see Figure 5.5). In this example, infrared sensors, DC motors, LEDs, and power systems were virtualized and modeled in the simulation.

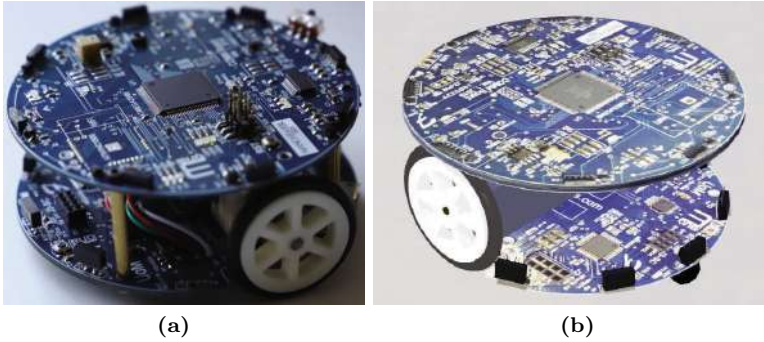


Figure 5.5: (a) Mona, an open-source, cost-effective robot designed for swarm robotics [17], and (b) a virtual model of Mona in Webots [11]. ↵

Another example of importing a new robot into the Webots software is shown in Figure 5.6(b). The Colias micro-robot, widely used for swarm robotic research, is modeled with great detail. The model is equipped with an RGB camera, three IR proximity sensors, and a power system similar to the one developed for the real robot. The detailed model shown in Figure 5.6(c) is mainly for presentation purposes; therefore, the simulator’s visualization function must be deactivated if we need to run experiments with a larger number of robots (> 25 robots) on a standard desktop computer.

Physics-based simulations are increasingly used in more complex decentralized swarm systems. Typically they are used in the form of a feasibility study before the swarm controller is tested with real-world robots. For example, in Hu et al. [169], a framework for a fault-tolerant Search

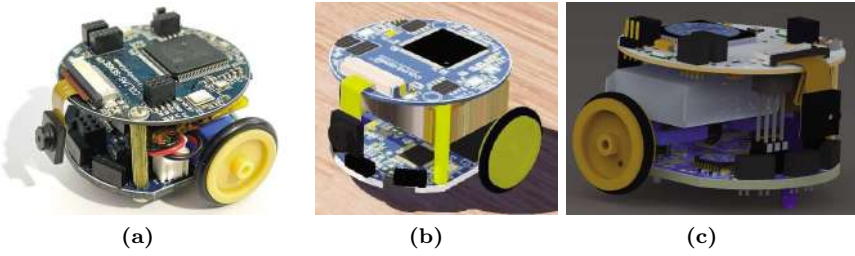


Figure 5.6: (a) Colias, an open-source micro-robot developed for swarm robotic applications [19], (b) a virtual model of Colias in Webots and (c) a detailed CAD model of Colias. ↵

& Rescue mission was built with a heterogeneous swarm system. They implemented a complex controller to steer the flock and an autonomous game-theoretic decision-making algorithm. It is obvious that developing such a complex system with a communication network and heavy computation directly on real robots is very challenging. Therefore, the physics-based simulation, in particular Webots, was chosen to test the proposed system by simplifying communication networks and tracking the system using a central observer. Figure 5.7 illustrates the scenario proposed in the Webots simulation. Other similar studies on formation control of multi-agent systems were presented in McCord et al. [241] and in Rekabi et al. [308] that simulated large multi-UAV systems in Gazebo. There are many more examples of multi-robot systems tested in physics-based simulations prior to real-robot implementations. In Aranda et al. [14], a formation control of a multi-UAV was simulated in Cobaye software [396], also a physics-based simulator. Another study in Baumann and Martinoli [30] proposed a modular framework for the navigation of a multi-robot system. The framework was implemented in Webots and the swarm robotics platform Khepera-IV. Moreover, for a multi-robot path planning system in a robotic manufacturing scenario, an optimal sequential task allocation was proposed in Brown et al. [47] which was simulated in Webots as well.

Another research direction in CPS swarms involves multi-vehicle and connected vehicle platoons. Recently, numerous studies have focused on the control challenges of multi-vehicle systems. These strategies are predominantly tested using physics-based simulations. For instance, vehicle fleet target tracking [352] has been simulated in CARLA [93], model predictive control [400] simulated in Prescan/Matlab/V2X, and Unreal Engine [128] has been utilized to simulate overtaking [413] and adaptive cruise control [415] in multi-vehicle systems. Figure 5.9 shows examples of multi-vehicle experiments using physics-based simulations.

Moreover, physics-based simulation are widely used for machine learning in swarm systems. Training a model using deep reinforcement learning is a time-consuming process. The system needs to run for thousands of iterations

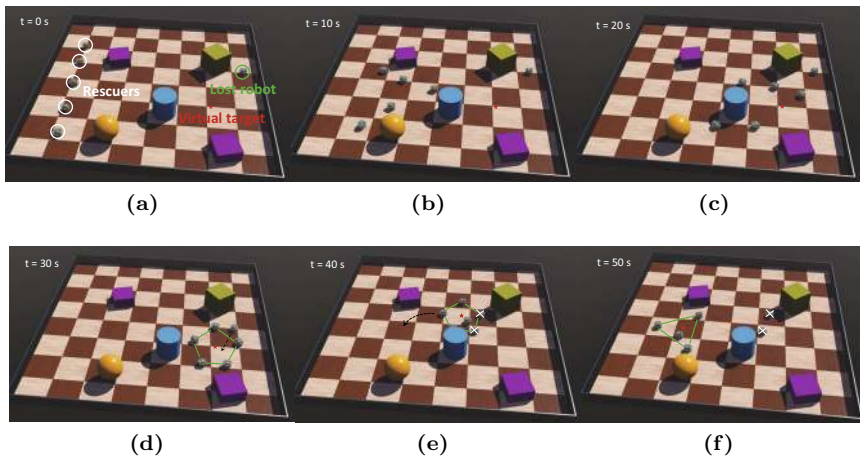


Figure 5.7: The Search & Rescue mission began at $t = 0$ s. Ten seconds later, rescuers were on their way to the virtual target. At $t = 30$ s, they had formed a pentagon around the target. By $t = 40$ s, the lost robot had been contained and was being guided home. Unfortunately, two of the rescuers (marked by White crosses) had stopped working due to an unexpected fault. At $t = 50$ s, the remaining rescuers had created a new triangular formation to guide the lost robot back to the starting point [169]. ◀

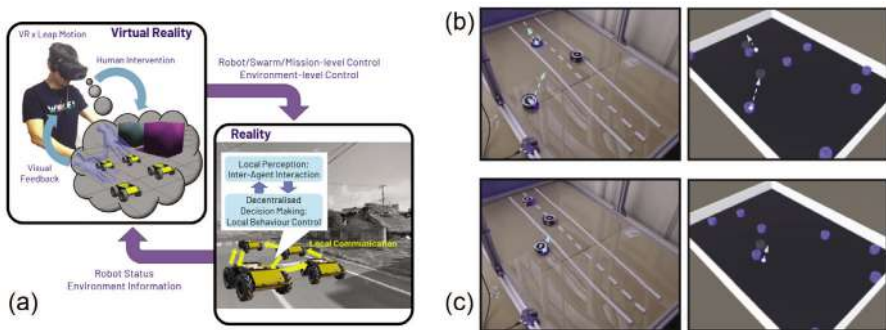


Figure 5.8: (a) The system architecture delineating the proposed human-swarm Interaction utilizing Omnipotent Virtual Giant, as detailed in Jang et al. [178]. Experimental validation showcasing (b) the relocation of holographic objects of robots and (c) the subsequent movement of real robots towards these relocated objects. Dashed arrows indicate the remaining trajectory to the target objects.

to train a good model. Therefore, developing models for swarm systems using real robots can be very difficult and almost impossible due to the autonomy time of the robotic platforms and their limitations in processing and memory units. Researchers tend to use physics-based simulation to train the model before they transfer it to real robots. As an illustration,

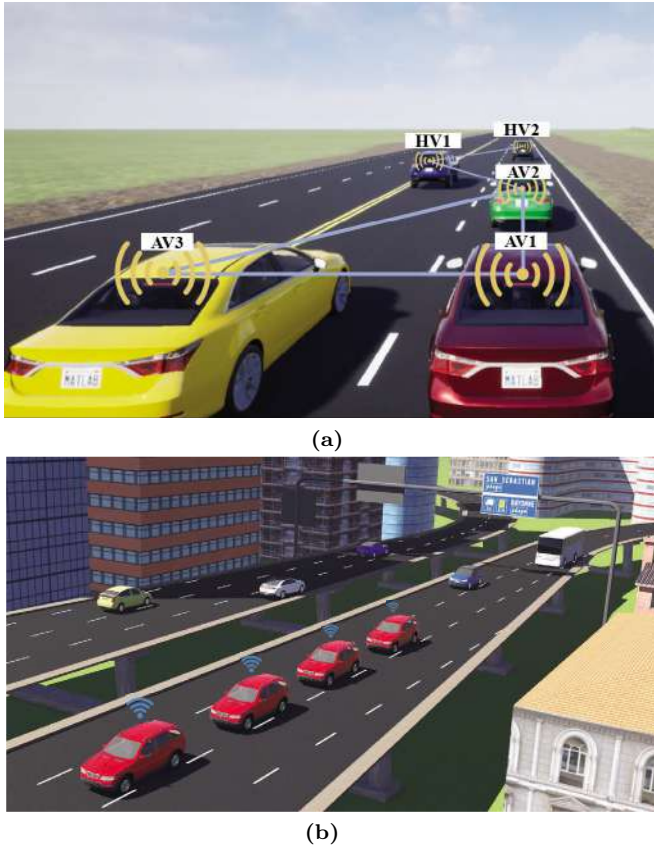


Figure 5.9: Screenshots of simulation examples of multi-vehicle simulations using (a) Unreal Engine [414] and (b) Webots [168]. ↵

Hu et al. [170] established an autonomous exploration task employing a multi-robot system. In this scenario, a model was trained through deep reinforcement learning within the Gazebo simulation environment. Figure 5.10 illustrates the training phase that has been done in Gazebo and real robot experiments with TurtleBot3 robots. A similar study proposed a collective navigation and obstacle avoidance system based on a federated reinforcement learning approach [268]. It was implemented with real robots, TurtleBot2, after the model was developed in Gazebo. In another study, a collision avoidance approach was developed in which robots were trained using deep reinforcement learning in Stage simulation [108]. A decentralized scenario with individual robots that generate paths considering a limited sensory system was investigated. There are many successful examples of using physics-based simulation for robotic swarm systems, such as human-swarm interaction to train artificial neural networks [75] and obstacle avoidance of robot swarms using a trained artificial pheromone system [265, 264].

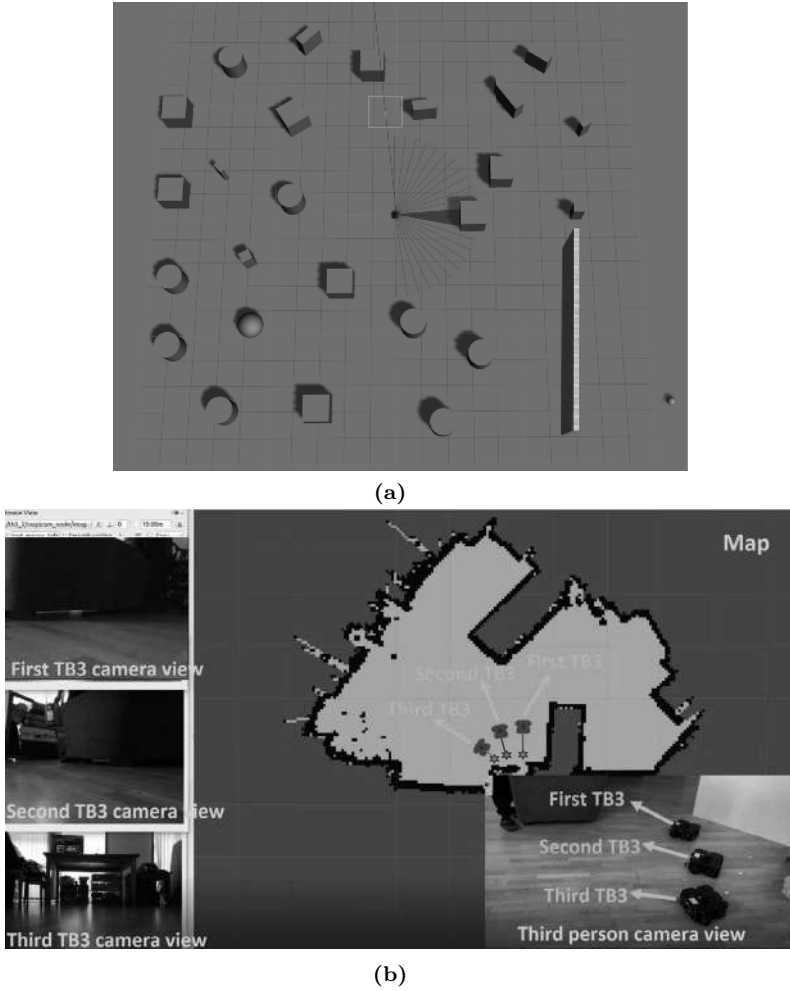


Figure 5.10: (a) Gazebo environment for training and (b) real-world experiment using three TurtleBot3 Waffle Pi mobile robots [170]. ↵

Physics-based simulations are also combined with real robots in the development of mixed-reality missions. In these settings, a larger population is usually developed in a simulation software, and a few of these robots represent real robots in a real-world setting. A recent work of Jang et al. [178] presents an interesting implementation of a mixed-reality human-swarm interaction where real robots are deployed in the presence of virtual robots in a bespoke developed Unity-based simulation platform. A human operator supervises a swarm by interacting with simulated robots [139]. An additional instance of mixed-reality human-swarm interaction is demonstrated in Patel et al. [284, 283], enabling human interaction with the swarm at both environmental and swarm levels.

5.3.1 BeeGround: A Simulation Platform

BeeGround [229] is an open-source simulation platform developed on the Unity Development Engine [127]. The main purpose of developing BeeGround was to make a fast and reliable swarm simulation accessible for everyone, with the capability of simulating swarm scenarios from a very abstract level to advanced detailed physical models. This allows researchers to increase and tune the level of modeling as needed. With Unity's physics engine and design interface, a plug-and-play package was developed, allowing researchers to implement a swarm scenario and environments of varying sizes, obstacle placements, and swarm populations to develop desired swarm controllers. Different sensors and other modules can be easily added or removed. The actuation mechanism can be altered, or the agent can be swapped out entirely. In addition, Unity supports TensorFlow 2.0 integration that expands the application of BeeGround for machine learning in swarm robotic systems. Figure 5.11 shows a swarm scenario developed using the BeeGround platform implementing a cue-based aggregation with a large swarm of Mona robots.

Before starting a simulation, we can customize the swarm's behavioral and physical parameters. We start by setting up an area where the size, shape, and presence of obstacles can be defined to create a variety of scenarios. For example, when using the cue-based bio-inspired algorithm, BEECLUST, an additional heat map can be loaded that is used by the agents to reference the temperature conditions within a region. The swarm parameters provide users with the capability to define key aspects such as population size, initial agent positions within the arena, and various kinematic constraints. Furthermore, integration packages for ROS-Unity facilitate the incorporation of well-established ROS projects into Unity, allowing for the seamless publication and subscription to topics between the two platforms. Detailed programming in Unity is not expounded upon here, given the extensive library of learning resources within the development engine and the availability of an active community for assistance and comprehension. Instead, we provide an overview of the components necessary to start and operate BeeGround.

Robot Modeling: Unity Engine scripts have two main components that form the basis of the robot's controller: A start function and an update function. The start function is analogous to the initialization phase of the robot and is executed at the start of the simulation. The update function is similar to a **while** loop and is continually executed throughout the simulation. The robot's behavior is determined within this update function.

We can define a robot by either importing a standard Unified Robotic Description Format (URDF) file or constructing one within the Unity Environment. Rigid body components can be added to the robot, which will be affected by the physics engine. This allows for the application of forces, torques, and collisions within the engine's fixed update function. Joints can also be implemented, which enables the creation of wheels and robotic limbs. Moreover, sensors such as cameras and range finders can be created using the tools provided by the Unity development engine.

Arena Configuration: The initial step in configuring the arena is to specify its size in standard units. BeeGround will then create a walled-off area with the given dimensions. Afterwards, the user can input an occupancy grid to place a cubic unit of obstacles in the environment. Furthermore, Unity provides other assets that can be used to create more intricate obstacles if desired.

Swarm Parameters: Bees, which are models of robots, can be created from the beginning with adjustable parameters, ranging from abstract to physics-based simulation. Instead of the Bee agent, custom robot models can be used for swarm generation. Furthermore, BeeGround allows the swarm population and placement to be specified, thus enabling some extraordinary testing scenarios.

Simulation Parameters: For ease of use, the length of a simulation and the number of repetitions can be specified in this section. Additionally, the speed of the simulation can be adjusted while it is running. This feature offers great versatility for running extended experiments with multiple repetitions that are necessary for statistically analyzing the outcomes.

Bio-inspired-specific Parameters: The environmental conditions, such as humidity, temperature, light, and more are essential for bio-inspired swarm robotics scenarios. To create a realistic simulation platform for robotics swarm applications, these environmental properties must be incorporated into the simulator. For example, a heat map was included in Wang et al. [401] to test the honeybee aggregation algorithm in a complex environment. This heat map was made up of an array of temperatures that the agents used as a reference for their waiting times. We can use multiple layers of these arrays to introduce environmental properties from different sources and models.

We can also create dynamic models of the environmental conditions that vary over the course of an experiment. These environmental characteristics can interact with the robots, for example, to create a bio-inspired pheromone communication system. To log swarm experiments, we record various parameters of the agents, such as position and orientation, every second to observe the behavior of the swarm over time. The output logs can be tailored to the user's requirements, as other parameters, such as velocities and rotations, are accessible through the Unity interface.

As previously mentioned, faster simulations can be achieved by reducing or deactivating the physics, although this comes at the cost of accuracy. This allows us to simulate swarm behaviors with thousands of robots, as illustrated in Figure 5.12. This capability of running experiments with very large swarm populations opens up new research directions that have not been explored before. For example, in Kiszli and Arvin [207], a bio-inspired cue-based aggregation was studied with a large swarm robotic system. It was found that the original goal of BEECLUST aggregation could not be generalized as a result of a barricade effect observed from experiments with large swarms.

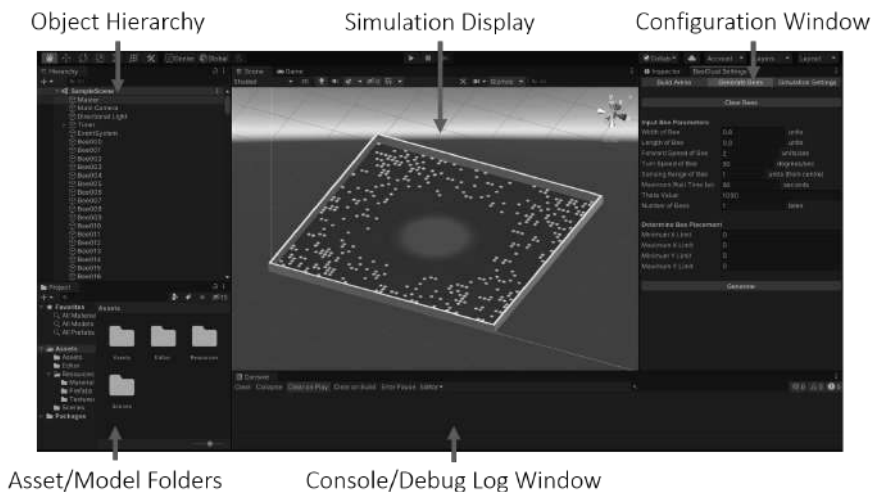


Figure 5.11: BeeGround UI featuring arena with a gradient cue with a large size swarm. The *Object Hierarchy* provides an overview of all the elements in the environment, such as robots, walls, obstacles, and cues. The *Simulation Display* is a graphical representation of the simulation. The *Configuration Window* allows users to adjust the settings for the BeeGround simulation, including the agents and their behavioral parameters. The *Asset/Model Folders* is a library of all the assets related to the simulation. Finally, the *Console/Debug Log Window* tracks all errors or debugging information that occur during the simulation [229]. ↵

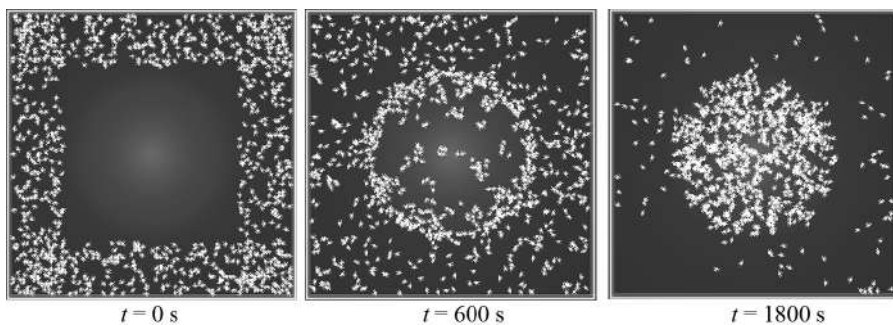


Figure 5.12: An example of BEECLUST aggregation in the BeeGround with 1000 robots in an environment with a single gradient cue (from [229]). ↵

Chapter 6

Swarm Robotic Platforms

Numerous robots have been developed for use as swarm robotic platforms, such as Spiderino [181] and Colias [19], each tailored for a particular task mimicking bio-inspired behaviors and scenarios. Additionally, various general-purpose mobile robots like e-puck [249] and Thymio [250], have been widely employed in robotic swarm applications. Swarm robotic platforms typically share common features such as small size, low-cost and a preference for open-source hardware and software. Furthermore, it is crucial to have versatile robots that are easy to program and modular, enabling the re-utilization of robots in multi-purpose tasks. We will examine the fundamental capabilities and explore the basic functionalities of a swarm robotic platform chosen for swarm applications.

6.1 Sensors

To interact with the environment and other robots, a reliable sensory system is essential for a swarm robotic system. Standard sensors such as short- and long-range IR and ultrasonic proximity sensors, as well as low-resolution cameras, are commonly used. Long-range IR sensors are especially popular because of their simplicity, as they operate by driving a photodiode to transmit and reading IR values from a phototransistor as a receiver. These sensors enable robots to identify other members of the swarm and distinguish themselves and others, from obstacles or other elements in the environment. This capability is important for the localization of the swarm [233]. The range of these sensors can vary from 10 cm to a few meters, depending on the IR radiance intensity (measured in mW/sr, milliwatts per steradian). Reflected IR from obstacles, such as walls or other objects follows the fundamental principles of

electromagnetic radiation and can be mathematically modeled as,

$$s(x, \theta) = \frac{\alpha \cos \theta}{x^2} + \beta, \quad (6.1)$$

where $s(x, \theta)$ represents the recorded value from the sensor, with x denoting the distance of the obstacle from the sensor, and θ indicating the angle of incidence with the surface. The variables α and β encompass various parameters, including the reflectivity coefficient, the output power of emitted IR, the sensitivity of the sensor, and the ambient light effect. These parameters are typically estimated empirically. Consequently, white and dark surfaces exhibit different ratios of reflection and absorption of IR radiations. This distinction is a critical consideration when devising an experimental setup for a swarm system. In swarm robotics, understanding the surrounding environment is a crucial requirement for a robot. They need to detect neighboring robots before an interaction starts; hence, several sensors are placed around a robot. As an example, Figure 6.1 illustrates the sensory reading of the *Colias* micro-robot that detected another robot in front. The varying intensity of the IR readings from multiple receivers enables the robot to estimate the relative position and orientation of neighboring robots using the following equation:

$$\phi = \text{atan} \left(\frac{\sum_{i=1}^{n_s} \hat{s}_i \sin(\gamma_i)}{\sum_{i=1}^{n_s} \hat{s}_i \cos(\gamma_i)} \right), \quad (6.2)$$

where ϕ is the estimated bearing of a neighbor, n_s is the number of sensors used by the robot, γ_i is the angular distance of the i th sensor with respect to the robot's head, and \hat{s}_i is the IR intensity received by the sensor i . By employing simple data analysis on the robot's microcontroller, the robot can detect neighboring robots and estimate their position and bearing.

In addition to IR proximity sensors, existing swarm robotic platforms are also equipped with various off-the-shelf sensors. They have been used as cue markers which measure, which measure ambient light illumination have been particularly popular. They have been used as cue markers in honeybee aggregation [329] and to simulate pheromone trails in ant-like exploration [18]. The more sophisticated version can also read RGB colors (as shown in Figure 6.2), which facilitates implementing multi-layer pheromone systems, as demonstrated by Liu et al. [231]. More details about the pheromone system and its implementation for swarm robotics will be discussed later in this chapter.

6.2 Actuators

Swarm robotic platforms employ various types of actuation mechanisms that have been used to actuate swarm robotic platforms. For instance,

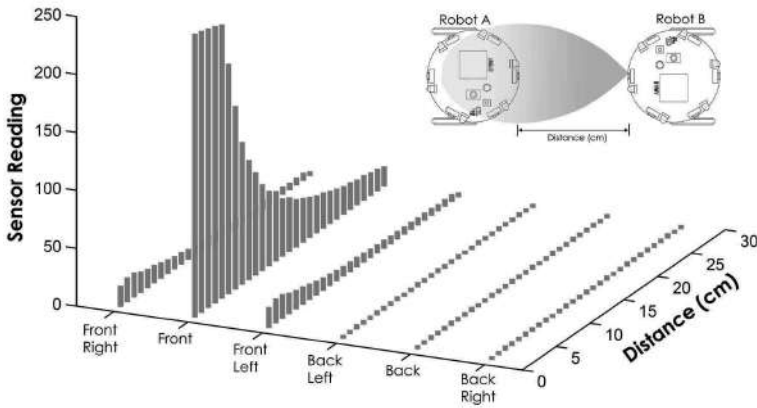


Figure 6.1: An example of a swarm robotic platform, equipped with six IR sensors, where Robot A receives IR from Robot B that has a 0° orientation (from [19]). ↵

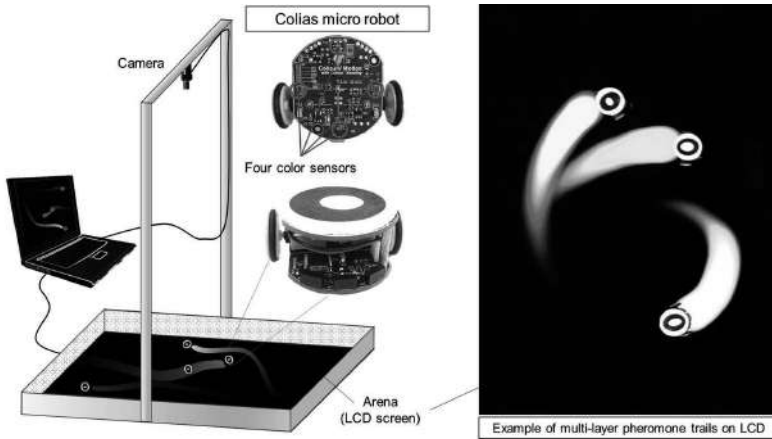


Figure 6.2: An example of a multi-layer pheromone system (from [231]). ↵

Spiderino [181] uses six legs for walking on the floor, Kilobots [316] move on smooth surfaces by vibrating their three legs, UAVs in the CoCoRo project [330] used thrusters for underwater swimming, and drones in Crazyswarm [293] use propellers for flight. However, due to their simplicity of control, wheeled robots are the most common actuation mechanism in swarm robotics. There are also hybrid mechanisms such as tracks and wheels developed by Mondada et al. [252], so called treels, that make navigation simpler and smoother.

In terms of wheeled robots, there are various types of wheels configurations such as differential drive (two and four wheels), car-type steering, omnidirectional and synchronous drive, among others. Due to the mechanical and control simplicity of the two-wheel robots, many swarm

robotic platforms have two wheels with a differential drive steering mechanism, which controls the trajectory of the robot by controlling the left and right wheel independently.

6.3 Communication

There are various types of communication used for swarm robotic platforms, direct communications using, e.g., IR, Wi-Fi, Bluetooth as communication technology to send specific messages, and implicit communication, communication over the environment, for example, pheromone-based. The direct form of communications is implemented by: (i) One-to-one, where a robot can directly talk to another robot and share information, (ii) one-to-many, where a robot can broadcast its messages to many individuals (leader to followers), (iii) many-to-one, where several robots communicate with an individual (followers to leader), and (iv) many-to-many, where swarm members communicate with each other without having a priority between them. As passing messages are always very specific to the underlying communication technology, we will focus on stigmergic, pheromone-based communication in the next section.

6.3.1 Pheromone-based Communication

Pheromones are a chemical substance that are detected by members of the same species, causing them to act in a certain way [194]. These substances play a vital role in facilitating communication among a diverse array of organisms, ranging from yeast and insects to mammals. Vertebrates have been discovered to use pheromone-based communication. Research has revealed that humans can experience physiological and psychological reactions due to chemosignals [263]. Many animals rely on pheromones as their main form of communication, but, to the best of our knowledge and the swarm perspective, social insects make the most out of this method. Pheromones enable extensive communication among a group of insects, creating a shared memory-like effect. Moreover, the utilization of pheromones serves to optimize the collective behavior of social insect groups. For instance, ants employ pheromones and feedback systems to efficiently locate the shortest route from their nest to a food source.

As an example of pheromone-based communication in social insects, *Monomorium pharaonis* called Pharaoh's ants, which are usually found in human habitats, use multiple types of pheromones that are important for their foraging behavior in dynamic and competitive environments [177]. For example, they generate a complex combination of pheromone trails between the food source and the nest using three distinct types of pheromones (i) non-volatile attractive pheromones, (ii) volatile attractive pheromones, and (iii) repellent pheromones. In another example, bumblebees, *Bombus hortorum*, leave chemical cues on flowers that allow detection and avoidance of recently depleted flowers [103]. Similarly, chemical cues enhance the efficiency of bee

colonies' foraging behaviors, as they prevent meaningless visits to depleted flowers. Another type of pheromones in nature are queen pheromones that characterize the queen and other members, which are essential to maintain the colony. For example, if the queen fails for any reason, including viruses and pesticides, the secretion of the queen pheromone in the colony decreases, causing the colony to collapse [365].

Pheromones in Robotics

Pheromone-based communication is one of the bio-inspired communication mechanisms widely used in swarm robotics. It requires a simple capability for an individual robot since it only needs local sensing. Additionally, the environment plays the role of memory; hence, individuals only need to have limited local memory. Moreover, we can optimize the performance of a swarm system using a combination of multiple types of pheromones and feedback mechanisms. And, most importantly, the system is fully decentralized without requiring a central control.

One of the first works inspired by pheromone communication was introduced by Russell [318]. This work used cinnamon camphora, known as Camphor, as the implementation of trail pheromones for trail following behaviors for robotic systems, which embodied odor release and detection functionalities. It showed the applicability of pheromone-based communication in robotic systems. In another research work [125], they used ethanol to simulate a pheromone for the robotic system.

Furthermore, numerous studies have suggested robotic systems that leverage Radio-Frequency Identification (RFID) tags as a medium for pheromonal communication [7, 159, 202]. RFID tags were attached to the floor where the robots operate. The tags store data transmitted by the robots passing above and eliciting the corresponding behavior depending on its nature. This principle works like pheromones in nature. Recent studies have involved virtual environments in implementing artificial pheromones for the communication of swarm robotic systems. As an illustration, Campo et al. [54] introduced a mechanism for path selection in a foraging robot swarm employing virtual ants. In this approach, robots engaged in local transmission and reception of messages. Although this work implemented virtual pheromones only within robots, the other works created a virtual map to mark the deposited pheromones accessible to all robots [307, 317]. The robots are outfitted with virtual sensors, accessing shared virtual environments among all robots, which incorporate overhead tracking and control. There are many interesting implementations of artificial pheromone communications, for instance, Kilogrid [391] developed for Kilobot micro-robot, Phormica [321] implemented with e-puck. Several researchers have implemented pheromone-based communications by various means. Garnier et al. [131] simulated pheromones by projecting light from a ceiling mounted video projector.

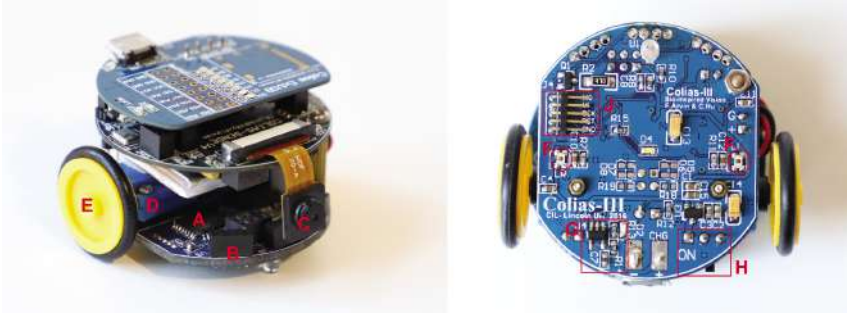


Figure 6.3: On the left is the Colias micro-robot and on the right is the Colias bottom board, featuring pheromone sensing capability. Various modules of Colias include A) main processor, B) IR proximity sensors, C) digital camera, D) micro-motors with gearhead, E) 22 mm wheels, F) pheromone detectors (light intensity sensors), G) battery recharging unit, H) main switch, and J) ISP programming port (from [266]). ↵

COS Φ is an open-source artificial pheromone system for swarm robotics that uses light-based pheromone trails [18]. The utilization of an LCD screen served as the arena for robots to engage with pheromones, visually represented as illuminated spots on the screen. Through a tracking system facilitated by a camera positioned above the arena, the system continually updated the robots' position, orientation, and ID, generating virtual pheromones in response. COS Φ offers several advantages: (i) It boasts a notably high resolution for pheromone implementation in comparison to other approaches, (ii) its highly flexible nature allows for the implementation of diverse environments, including variations in pheromone trail thickness, evaporation rate, and diffusion, and (iii) it employs a cost-effective configuration featuring a basic digital camera [212]. A flat LCD screen enhances the accessibility for a wide range of researchers. The Colias microrobot [19], shown in Figure 6.3, was deployed for the implementation of artificial pheromone communication. It has been extended in several studies [267, 231, 266] to include diffusion, wind effect, multi-layer pheromones, which are phenomena almost ever present in nature. Figure 6.4 illustrates a cue-based aggregation scenario in the presence of pheromones implemented by an artificial pheromone communication.

Artificial Pheromone System

The artificial pheromone system COS Φ comprises two integral components: A pheromone system and a tracking system. The pheromone system calculates and displays the pheromone on a flat LCD screen that is placed horizontally, representing the environment in which the robots are operating [18]. The tracking system tracks the robots and sends their data to the pheromone system [212]. An example of an artificial pheromone environment with real-world mobile robots is shown in Figure 6.5.

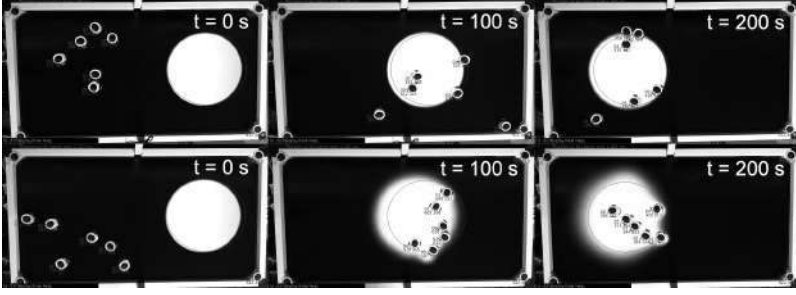


Figure 6.4: The first row presents sample experiments conducted without diffusion and with fast cue speed, devoid of pheromone injection. In contrast, the second row illustrates sample experiments conducted with diffusion and fast cue speed, incorporating pheromone injection, captured at time instances $t = 0$ s, $t = 100$ s, and $t = 200$ s from left to right (from [266]). ↵

The pheromone system models multiple types of artificial pheromones and their interactions simultaneously. The brightness of a grayscale image is represented as \mathbf{I}^1 , which is a two-dimensional matrix with the size of the resolution of the LCD screen. It is determined by Φ , which is a two-dimensional matrix of the same size as \mathbf{I} that represents the intensity of the pheromone. Each element of \mathbf{I} is equivalent to the brightness of the corresponding pixel. The brightness of the image at position (x, y) , $\mathbf{I}(x, y)$, is expressed with

$$\mathbf{I}(x, y) = \sum_{i=1}^n c_i \Phi_i(x, y). \quad (6.3)$$

The intensity of the i^{th} pheromone at position (x, y) is represented by $\Phi_i(x, y)$, and c_i is the influence of the i^{th} pheromone on the screen. $\mathbf{I}(x, y)$ is calculated by multiplying $\Phi_i(x, y)$ and c_i and summing them up, which means that n pheromones can be overlapped. To illustrate how the model works, the combination of three different pheromones with different effects on the screen can be displayed on a single-pixel after the calculation of $\mathbf{I}(x, y)$ using Equation 6.3.

During the operation of the system, the amount of pheromones released on the screen is continuously adjusted at discrete intervals. The intensity of the modified pheromone is expressed with

$$\begin{aligned} \Phi_i^{k+1}(x, y) = & -\mathbf{u} \cdot \text{grad} \Phi_i^k(x, y) - \frac{\ln(2)}{e_i \Phi} \Phi_i^k(x, y) \\ & + \kappa_i \nabla^{\text{grad}} \Phi_i^k(x, y) + \iota_i(x, y). \end{aligned} \quad (6.4)$$

In the given expression, $\Phi_i^{k+1}(x, y)$ denotes the intensity of the i^{th} pheromone at discrete time $k + 1$, while $\Phi_i^k(x, y)$ represents the intensity

¹Bold font parameters in this section indicate two-dimensional vector.

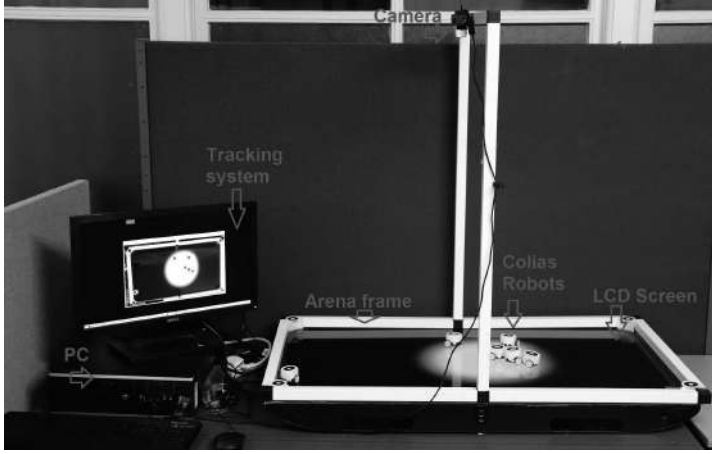


Figure 6.5: The experimental setup employed for the pheromone system in Na et al. [266] encompasses a PC dedicated to tracking robots and generating pheromones, a digital camera for monitoring robot positions, a horizontally positioned 42" LCD screen, an aluminum frame encircling the arena, and the presence of *Colias* mobile robots. ◀

of the i^{th} pheromone at discrete time k . The quantity $grad\Phi_i^k(x, y)$ is a two-dimensional vector describing the gradient of pheromone intensity at the position (x, y) and is defined mathematically by Equation 6.5. The symbol \mathbf{u} signifies the velocity vector responsible for linearly shifting the pheromone across the arena. The term $e_{i\Phi}$ governs the evaporation rate of the i^{th} pheromone and is characterized by its half-life, κ_i denotes the diffusion constant of the i^{th} pheromone, and $\iota_i(x, y)$ corresponds to a newly injected pheromone at the position (x, y) on the screen. The spatio-temporal development pheromone intensity model is derived from a simplified version of the Navier-Stokes equation, representing the fluid flow model.

$$grad\Phi_i^k(x, y) = \frac{\Phi_i^k(x+1, y) - \Phi_i^k(x-1, y)}{2} \mathbf{i} + \frac{\Phi_i^k(x, y+1) - \Phi_i^k(x, y-1)}{2} \mathbf{j}. \quad (6.5)$$

By recalculating Equation 6.4 for each value of (x, y) , it is possible to compute new pheromone intensities from their prior states, which consequently determine the new pixel values of the grayscale image displayed on the screen.

The parameters on the right-hand side of Equation 6.4 can be divided into two categories: Environmental effects and pheromone injection. Environmental effects, such as evaporation rate $e_{i\Phi}$, diffusion constant κ_i , and velocity vector \mathbf{u} (where i is omitted to generalize), have a constant influence on the pheromone released in the arena while the system is running. In contrast, the injection of pheromone $\iota(x, y)$ only affects the intensity of

pheromone in the arena when certain conditions are met, such as when robots stop. Under these conditions, the pheromone is injected in a circular shape with a given intensity. The injection of pheromone $\iota_i(x, y)$ is defined as

$$\iota_i(x, y) = \begin{cases} s_\Phi, & \text{if } \sqrt{(x - x_r)^2 + (y - y_r)^2} \leq l_\Phi/2 \\ 0, & \text{otherwise} \end{cases} \quad (6.6)$$

where (x_r, y_r) respectively represent coordinates of a robot and (x, y) the position of the pheromones in the arena, s_Φ is the intensity of injected pheromone at a time, and l_Φ is the diameter of injected pheromone. In a circle with a diameter of l_Φ , where the center of the circle is positioned at the robot's coordinates (x_r, y_r) , pheromone is uniformly injected at a rate of s_Φ .

Environmental Effects on Pheromones

The versatility of the system is enhanced by environmental effects, which enable it to replicate realistic conditions that influence the spread of pheromones in the environment over time.

Evaporation is the transformation of a liquid's surface into a gas. Volatile chemicals, such as pheromones, evaporate as well. In various studies on the kinetic properties of pheromones, the half-life of the pheromone is used as a measure [398]. This is the amount of time it takes for the pheromone to reduce by half, denoted as $e_{i\Phi}$ in Equation 6.4.

Diffusion is the process by which molecules move from a region of higher concentration to a region of lower concentration. Incorporating diffusion into the pheromone system is essential for realism, as the diffusion of pheromones significantly influences swarm behavior. Therefore, instead of using the mathematical definition of diffusion, $\kappa_i \nabla^2 \Phi_i(x, y)$, Gaussian blur is employed to simulate diffusion. This approach has the benefit of being able to replicate faster diffusion with fewer computational resources. The intensity of pheromone at the position (x, y) after application of the Gaussian blur is given by

$$\Phi_i^{k+1}(x, y) = (\omega * \Phi_i^k)(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b \omega(s, t) \Phi_i^k(x - s, y - t), \quad (6.7)$$

where $\Phi_i^{k+1}(x, y)$ is the intensity of the i^{th} pheromone at the discrete time $k + 1$, $\Phi_i^k(x, y)$ is the intensity of the i^{th} pheromone at the discrete time k and ω is a two dimensional kernel matrix with a size of $(2a + 1) \times (2b + 1)$ defined as

$$\omega(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \quad w \in \mathbb{R}^{2a+1 \times 2b+1}, \quad (6.8)$$

where σ is a standard deviation of the Gaussian distribution of the kernel matrix. Equation 6.8 indicates that the elements of ω are determined by

the Gaussian distribution. This blur technique is advantageous due to its computational efficiency, as well as its capacity to provide more intuitive control of the diffusion rate and area. Integrating Gaussian blur into the pheromone system aligns with diffusion properties for two key reasons in the context of our study: (i) Gaussian blur results in a decrease in the higher intensity of pheromone and an increase in the lower intensity, and (ii) the total pheromone quantity remains constant following each computational step.

Pheromone Shift changes the position of the released pheromone in real and dynamic settings. Advection, which is the movement of any fluid, such as air, that carries pheromone from one place to another, is a natural cause of this shift. The flow of the released pheromone in the same direction is represented by $\mathbf{u} \cdot \nabla \Phi(\mathbf{x}, \mathbf{y})$. The two-dimensional velocity vector is expressed with

$$\mathbf{u} \cdot \nabla \Phi_i(x, y) = u_x \cdot \frac{\partial \Phi_i(x, y)}{\partial x} + u_y \cdot \frac{\partial \Phi_i(x, y)}{\partial y}, \quad (6.9)$$

where u_x, u_y respectively represents the speed along x-axis and y-axis.

The use of pheromones in swarm systems as a communication mechanism offers scalability of collective behaviors without the need for direct communication between agents. For example, a single ant and a large swarm of ants can use the same pheromone trail to forage without increasing communication costs. This technique has found application in diverse research domains, including optimization [87], vehicle routing [361], and robotics [54, 391, 321, 266]. However, there are still some challenges to overcome before pheromone-based communication can be widely adopted, such as designing controllers for individual agents that can maximize the performance of the entire swarm [91].

Swarm robotic systems that employ pheromone-based communication demonstrate advanced collective behaviors suitable for real-world applications. For example, in Alfeo et al. [7], a vehicle swarm effectively manages waste in a simulated urban setting, surpassing traditional methods. Furthermore, pheromone-based communication is proposed as an optimal solution to manage large-scale autonomous vehicle traffic with a focus on avoiding collisions [265]. Such systems have also been adapted for urban construction scenarios [9]. Beyond these studies, numerous other practical applications of pheromone-based communication are emerging, where complex coordination of large-scale swarm robotic systems are required.

6.4 Swarm Robotic Research Platforms

The collective behavior of a swarm emerges from interactions between simple robots that simultaneously has an indirect relationship with the behavior of each individual robot of the swarm. It is observed that a simple modification of an individual's behavior (such as velocity and sensitivity) could result in a significant positive or negative impact on the collective behavior of the swarm

system. These can happen due to both hardware and software heterogeneity. The robotic platforms developed for a swarm system must be able to reliably imitate swarm behaviors which are mainly found in nature, such as birds flocking [385] and honeybees aggregation [329]. The robots must be designed with a compact size allowing implementation of large-scale swarm behaviors in the laboratory settings at low cost. To implement a variety of swarm behaviors on the individual robots, the design of the the robotic platform mechatronics must be versatile up to a specific level to (i) simplify replication and (ii) ensure platform homogeneity.

In the subsequent sections, we present a compilation of swarm robotic platforms utilized in research and education (refer to Table 6.1). These platforms have been specifically designed for swarm robotics applications or are commonly employed to implement swarm behaviors. The table is structured as follows: Entries are categorized based on the environment in which they operate, terrestrial, aerial, aquatic, and/or space. Each entry includes information on the type of application, the project or product name, the robot type, the swarm size (typically reflecting the number of robots used in the referenced evaluation), and the fundamental swarm behaviors corresponding to the definitions outlined in Section 4.1. We differentiate between various types of swarm robotic platforms including UGV, UAV, Unmanned Surface Vehicle (USV), or Unmanned Underwater Vehicle (UUV). The table and the explanation is an updated and extended version from Schranz et al. [337]. Be aware that other, probably more sophisticated robotic research platforms exist, which are not included. In this table and corresponding sections we focus only on platforms developed with the intention of using them in swarm robotic applications, specifically for research and education.

6.4.1 Terrestrial

The probably best known swarm of robots is the Kilobots swarm [316]. Kilobots have a diameter of only 33 mm, they move with the help of vibration motors, and the communication is implemented with IR light mounted on the bottom of the robot to reflect off the ground. Their most famous application is self-assembling, where they form different shapes using 1,024 Kilobots [412]. The Kilobot is available open-source² or commercially at K-Team³.

Another widely used swarm robotic platform is Jasmine. Also this platform is available open-source⁴. It was mainly built for large-scale swarm robotic experiments, where each robot is equipped with a set of sensors for touch, proximity, distance, and color. A similar aim regarding large-scale swarms was also actively pursued with the swarm robotic platform Alice [55]. A bunch of additional sensors, such as linear cameras extend the

²Kilobot website: <http://www.kilobotics.com/> [Online; accessed 26-March-2023].

³K-Team website: <https://www.k-team.com/> [Online; accessed 26-March-2023].

⁴Jasmine website: <http://www.swarmrobot.org/> [Online; accessed 26-March-2023].

basic capabilities. A series of research platforms building upon each other is given with AMiR [20], Colias [19] (open-source⁵ and commercially⁶ available), and Mona [24] (open-source⁷ and commercially⁸ available). The platform R-One [242] is also designed for usage as a swarm robotic platform. As a support for close to swarm intelligence experiments it uses a camera tracking system for ground-truth localization and a server-side located software connecting all the pieces. The Elisa-3 swarm robotic platform, open-source and commercially⁹ available, uses an Arduino microcontroller, including a large variety of sensors including eight IR proximity sensors, three-axis accelerometer, and four ground sensors. The swarm robotic platform has the capability to recharge autonomously using a charging station. The communication in the swarm is performed either via IR or radio. The Khepera IV [355] is designed for any indoor lab application. The technical features including a Linux core, color camera, WLAN, Bluetooth, USB Host, accelerometer, gyroscope, microphone, loudspeaker, three top RGB LEDs, and improved odometry makes this swarm robotic platform a compact and complete research platform for swarms in different scenarios. The Khepera IV is commercially available at K-Team¹⁰. The GRITSbot [288] is the open-source¹¹ swarm robotic platform used in the Robotarium¹² at Georgia Tech, Atlanta. The Robotarium is a physical environment that provides remote access to a large swarm of robots. Scholars, researchers and anybody interested in this topic can upload code to run experiments remotely. Coming with features like automatic registration of robots with a server, autonomous charging, wireless code upload to the robots, and automatic sensor calibration makes the Robotarium attractive for remote research experiments. The swarm robotic platforms in the Robotarium use wheeled locomotion and are equipped with a set of different sensors, for example, distance and light sensors.

The e-puck robot [249], together with its successor e-puck2, represent an educational and research robot designed to ease programming and control of a robot's behaviors. It uses diverse sensors, such as IR proximity sensors, a CMOS camera, and a microphone. The e-puck is available open-source¹³ or

⁵Colias open source website: <https://github.com/Farshad-Arvin/Mona-Platform/blob/master/Mona-Test.ino> [Online; accessed 26-March-2023].

⁶Colias commercial website: <http://www.visomorphic.com/> [Online; accessed 26-March-2023].

⁷Mona open source website: <https://github.com/MonaRobot> [Online; accessed 26-March-2023].

⁸Mona commercial website: <https://ice9robotics.co.uk/> [Online; accessed 26-March-2023].

⁹Elisa-3 website: <https://www.gctronic.com/doc/index.php/Elisa-3> [Online; accessed 26-March-2023].

¹⁰K-Team website: <https://www.k-team.com/> [Online; accessed 26-March-2023].

¹¹GRITSbot website: https://github.com/robotarium/gritsbot_2 [Online; accessed 26-March-2023].

¹²Robotarium website: <https://www.robotarium.gatech.edu/> [Online; accessed 26-March-2023].

¹³e-puck open source website: <http://www.e-puck.org/> [Online; accessed 26-March-2023].

commercially at GCTronic¹⁴. An extension of the e-puck presents the Xpuck. It has additional aggregated raw processing power (as used in modern mobile system-on-chip devices) of two teraflops on board. Thus, higher-individual robot computation can be achieved, for instance, image processing using the ArUco Marker tracking [187]. The Thymio II robot [312] is built on understanding of programming and robotic concepts by using a wide range of sensors, e.g., temperature, IR distance, accelerometer, and microphone. It supports both, visual and text-based programming using Blockly. The Thymio II is available both open-source and commercially¹⁵. Another platform for open-source¹⁶ swarm robotics development for education and research purposes is called Pheeno [406]. Custom modules allow the adaption of the platform according to the user's needs. Using IR sensors, it interacts with the environment. The Spiderino platform [181] is a six-legged open-source¹⁷ robot with spider-like locomotion. The basis forms a hexpod toy that is additionally enhanced with a PCB on top. This Printed Circuit Board (PCB) includes an Arduino microcontroller, a WLAN module, and several reflective IR sensors to allow robot control.

In the project I-Swarm (Intelligent Small-World Autonomous Robots for Micro-manipulation) the goal is to develop micro robots to form a swarm. The robot has a small size of only $3 \times 3 \times 3 \text{ mm}^3$, it is battery-free powered with solar energy, uses vibration motors to perform locomotion, and IR transceivers for communication [347]. The goal was to build a swarm of 1000 robots [347]. The robot's prototypes are exhibited in the technical museum in Munich, Germany.

The idea behind the open-source swarm robotics platform Zooids¹⁸ is different: It handles both the interaction among the robots and the display/interaction to human operators. Thus, it offers a new class of human-computer interfaces. The control of the swarm is performed through light patterns projected from the top using an overhead projector [223]. The APIS¹⁹ (Adaptable Platform for Interactive Swarm) has multiple components: The swarm robotic platforms, the test environment for the swarm including the necessary infrastructure, and the simulation framework [82]. The focus is on experiments related to human-swarm interaction. For such interactions, the robotic platforms are equipped with an additional OLED display and a buzzer. The Wanda [201] platform follows a special composition that could be used, for instance, to clean up the environment with a swarm. In addition, the authors implemented a robot-specific tool chain from design, simulation to

¹⁴e-puck commercial website: <https://www.gctronic.com/e-puck.php> [Online; accessed 26-March-2023].

¹⁵Thymio website: <https://www.thymio.org> [Online; accessed 26-March-2023].

¹⁶Pheeno website: <https://discourse.ros.org/t/pheeno-a-low-cost-ros-compatible-swarm-robotic-platform/2698> [Online; accessed 26-March-2023].

¹⁷Spiderino website: <https://spiderino.nes.aau.at> [Online; accessed 26-March-2023].

¹⁸Zooids website: <https://github.com/ShapeLab/SwarmUI> [Online; accessed 26-March-2023].

¹⁹APIS website: <https://github.com/wvu-irl/reu-swarm-ros> (software only) [Online; accessed 26-March-2023].

deployment. Another wheeled robot is the Mechalino, that will be available open-source under Github²⁰. Mechalino is a robot built from 3D-printed and commercial-off-the-shelf components. It is small and low-cost, but also very modular concept, allowing for extending the platform with extra sensors, actuators or computational capabilities if needed.

In the swarm logistics domain the authors in Jones et al. [186] built the DOT, a swarm robotic platform that is remotely accessible and comes with a complete toolchain to develop, simulate, and deploy code on the platform. In addition, to 5G and multiple cameras, it comes with a lifting platform on top. The WsBot [230] is another experimental platform for Industry 4.0 applications with a wireless charge system. It is mainly designed with forklifts to achieve intelligent behaviors individually or in groups, where all agents are interconnected over Wi-Fi. The Milli-Robot [167] is equipped with magnetic hardware with up to 288 degrees of freedom and is proposed for general swarm robotic research activities. The authors introduce random dithering and argue to achieve a 100% success rate (i.e., no deadlocking). Another swarm robotic platform is the mROBerTO 2.0 [106] limited in size equipped with a locomotion mechanism that utilizes stepper motors to focus on complex trajectories. This allows the robot to micro-step down to 1/32 of a full step.

The Droplet [208] is a swarm robotic platform for teaching and research. The spherical robot is able to organize and cluster into complex shapes with its neighbors. They use vibration motors for their locomotion. A specialty is that the robotic platforms charge and communicate via a powered floor that is equipped with alternating stripes of positive charge and ground. It is available as an open-source project²¹. The Swarm-bots [253, 149] can assemble themselves to different geometric 3D shapes. The robots are built by a number of simpler, insect-like robots that use relatively cheap components (the design is open-source²²). These robots can perform self-assembly and self-organization to adapt to the current situation in their environment. This assembling capability allows the robots in swarm formation to transport objects that would be too heavy for an individual robot. The follow-up project of the Swarm-bots is the Swarmanoid project. This project counts as the very first attempt to study the integration of design, development, and control of a swarm of heterogeneous, open-source²³ swarm robotic platforms. The swarm in the Swarmanoid project presents three different types of autonomous robots, each equipped with various sensors: i) Eye-bots are UAVs that have the ability to attach to an indoor ceiling, ii) hand-bots are UGVs that are able to climb, and iii) foot-bots are UGVs that have a self-assembling capability [89]. The ARGroHBotS [277] is another swarm robotic platform where the user can adjust parts of the hardware (mainly

²⁰Mechalino Github page: https://github.com/smartgrids-aau/Mechalino_Arena [Online; accessed 26-March-2023].

²¹Droplet website: <https://code.google.com/archive/p/cu-droplet/> [Online; accessed 26-March-2023].

²²Swarm-bots website: <http://www.swarm-bots.org/> [Online; accessed 26-March-2023].

²³Swarmanoid website: <http://www.swarmanoid.org/> [Online; accessed 26-March-2023].

3D printed pieces) to set it up for specific implementations. Additionally, the platforms allow remote access. The Termes robots [286] use modular blocks and create large structures without communication or GPS localization. They utilize the concept of stigmergy that is inspired by the way living termites build their nests. The Termes robots themselves have the abilities to climb and carry blocks to create these structures in unstructured, rough environments. Symbrion and Replicator [199] are two sibling projects, that develop autonomous swarm robotic platforms. They can be used as single entities or can be physically connected to form special patterns. The main goal of these projects was to focus on the possibility to achieve evolvability of robot organisms. PolyBots [96] are self-reconfigurable robots. They have various types of locomotion options and object manipulation modules that are interchangeable to form a number of shapes, for example, an earthworm type to slither through obstacles, or a spider to stride over hilly terrain. These robots find application especially in unknown environments or multiple task challenges. Further modular robots that allow self-configuration with similar robotic technologies include M-TRAN [261], M-TRAN II [217] and M-TRAN III [218] (available as open-source²⁴ project), ATRON [46], CONRO [57], Sambot [403], Molecube [429], Puzzlebot [420], to name a few²⁵.

Finally, a remarkable new field of the application of swarm robotic platforms is the acoustic swarm [176] that creates speech zones representing a self-distributing wireless microphone array. In their experiments the authors showed that their swarm is able to localize and separate three to five concurrent speech sources without the usage of external infrastructure (such as a camera system).

6.4.2 Aerial

There are already multiple miniature UAVs available that form a good basis for research in aerial swarm robotic systems. A comprehensive overview of such small-scale UAVs can be found in Cai et al. [50] and Swetha et al. [373]. Off-the-shelf Micro Air Vehicles (MAVs) are available in a wide range of models and are widely used in the gaming industry, and other businesses for video and photography. Unfortunately, their flight controllers are typically closed so that we cannot develop and test custom algorithms (e.g., Qualcomm Flight Pro²⁶, DJI M100²⁷). A UAV that is designed specifically for usage in swarms is the MAV presented by Roberts et al. [313]. These UAVs are equipped with three rate gyroscopes, three accelerometers, one ultrasonic sensor, and four IR sensors. It has been co-developed in the Swarmanoid project [89]. In the

²⁴M-TRAN website: <https://www.wevolver.com/wevolver.staff/m-tran> [Online; accessed 26-March-2023].

²⁵For a full list, the reader is referred to https://en.wikipedia.org/wiki/Self-reconfiguring_modular_robot [Online; accessed 26-March-2023].

²⁶Qualcomm Flight Pro website: <https://www.lantronix.com/products/qualcomm-flight-pro-development-kit/> [Online; accessed 26-March-2023].

²⁷DJI M100 website: <https://www.dji.com/at/matrice100/info\#specs> [Online; accessed 26-March-2023].

Distributed Flight Array [280] each UAV makes up a module of a larger array, but has only a single rotor. The individual modules are able to self-assemble to form a multi-rotor system, where all vehicles must cooperate constantly for a coordinated flight. To achieve this behavior, they exchange information among each other and, using the neighbor's information, they adjust local parameters. The Crazyflies, available both open-source and commercially²⁸, are used to realize a swarm of UAVs indoors. They are equipped with multiple sensors, e.g., accelerometer, gyroscope, magnetometer, and a high precision pressure sensor [293]. They have quite a low weight of 27 g that reduces danger for humans during the experiments. For the localization the Crazyflies need an external tracking system such as OptiTrack²⁹. Another indoor swarm can be build with the FINken-III [158]³⁰ and its predecessors. They are equipped with optical flow, IR distance, and a tower of four sonar ranging sensors.

6.4.3 Aquatic

In the CoCoRo (Collective Cognitive Robotics) project [330] a huge swarm of 41 heterogeneous UUVs has been developed. For the experiments they use three types of vehicles: A base station USV, an exploration UUV, and a UUV for relaying information between the base station and the explorers. Communication is achieved using sonar and electric fields. The primary envisioned applications include environmental monitoring, assessing water pollution and evaluating the effects of global warming. The UUV Monsun [278] uses two different types of communication: An acoustic underwater modem for information exchange and a camera to recognize and follow other swarm members. The CORATAM (Control of Aquatic Drones for Maritime Tasks) [62] project develops swarms of USVs for future environmental monitoring, sea life localization, and sea border patrolling. The USV platforms are available open-source³¹ and execute swarm algorithms generated using evolutionary computation [94]. Another project is concentrating on the development of indoor USVs specifically the microUSV [146] which measures just 23 cm in length. They are still in the early stages and are currently capable of following waypoints. Nevertheless, indoor USVs for swarm robotics application are really necessary to perform behavioral deployment and research at low cost.

²⁸bitcraze Crazyflies website: <https://www.bitcraze.io/crazyflie-2-1/> [Online; accessed 26-March-2023].

²⁹Optitrack website: <https://optitrack.com/> [Online; accessed 26-March-2023].

³⁰FINken website: <https://www.ci.ovgu.de/SwarmLab/Robots/FINKens.html> [Online; accessed 26-March-2023].

³¹CORATAM website: <http://biomachineslab.com/projects/control-of-aquatic-drones-for-maritime-tasks-coratam/> [Online; accessed 26-March-2023].

6.4.4 Outer Space

The NASA developed the Swarmies³² for space exploration. The main goal is to collect material samples such as water, ice, or useful minerals on Mars. This application is referred to as in-situ resource utilization. Simultaneously, NASA launched a swarmathon³³ to motivate students in the development of swarm algorithms based on ant foraging. In the experiments 20 Swarmies can travel a linear distance of 42 km in 8 hours. This is the same distance that was covered by Mars rover Opportunity in 11 years. Another innovative project was accepted by the NASA Innovative Advanced Concepts program, namely the Marsbees [192]. Their objective is to enhance Mars exploration using flying swarm robotic platforms. They have the size of a bumblebee and self-explore the environment. The Mars rover Opportunity serves as base and charging station.

6.5 Project: How to Build your own Swarm Robot

6.5.1 A Robot with Legs: Spiderino

The Spiderino robot is an example of a legged robot that is aimed at swarm robotic experiments and educational purposes. Two major challenges for building a swarm robot, cost and size, are particularly difficult to master when it comes to legged robots. For instance, if a leg is equipped with two servo motors, one for lifting the leg and the other for moving it forward or backward, a robot with six legs would require several servo motors to operate and control. One possibility would be reducing the number of legs, but that would complicate the movement and possibly require extra sensors for the control system. For example, biped walkers require sensory systems to keep balance while moving over unknown ground. When aiming at a system which is statically stable at all stages of their stride, at least four legs are required. A quadruped following this principle would lift only one leg at a time and select the sequence of movements so that the center of gravity is always above the area spanned by the feet on the ground. Turtles are frequently mentioned as an example of a gait that maintains static stability. However, in reality, a turtle's stride can also include phases where the body is supported only by two diagonally opposite legs [6, 428]. For a sixed-legged robot, an effective gait involves lifting three legs at a time while maintaining stability with the other three legs. This approach allows for faster movement compared to the statically stable one-leg-at-a-time gait used by quadrupes. Therefore, the challenge remains: How can we design the mechanics in a cost-efficient and compact way.

³²Swarmies website: <https://www.nasa.gov/content/meet-the-swarmies-robotics-answer-to-bugs> [Online; accessed 26-March-2023].

³³Swarathon website: <https://swarmathon.cs.unm.edu/> [Online; accessed 26-March-2023].

Table 6.1: Classification of research platforms for swarm robotics (adapted from Schranz et al. [337]).

Environment	Application	Project / Product Name	Robot Type	No. of Robots	Basic Swarm Behaviors	Availability
Terrestrial	Research and Education	Kilobots	UGV	1024 [316]	pattern formation, coordinated motion	open-source, commercial
		Jasmine		60 [198]	aggregation, collective exploration, coordinated motion, task allocation, collective perception, self-healing (partially), human-swarm interaction (Zoids, APIS)	open-source
		Alice		20 [132]		n.a.
		AMiR		6 [21]		open-source, commercial
		Colias		14 [19]		n.a.
		Mona		30 [17]		open-source, commercial
		R-One		n.a.		n.a.
		Elisa-3		38 [137]		open-source, commercial
		Khepera IV		10 [289]		commercial
		GRITSbot		100 [287]		open-source
		E-Puck		16 [8]		open-source, commercial
		Xpuck		16 [187]		n.a.
		Thymio II		8 [387]		open-source, commercial
		Pheeno		4 [405]		open-source
		Spiderino		n.a.		n.a.
		I-Swarm		n.a.		open-source
		Zoids		32 [223]		n.a.
		APIS		6 [82]		open-source
		Wanda		11 [200]		n.a.
		Mechalino		4		open-source
		DOTS		20 [186]		n.a.
		WsBot		2 [230]		open-source
		Milli-Robot		16 [167]		n.a.
		mROBERTo		n.a.		n.a.
		Droplet		n.a.	aggregation, self-assembly, object clustering and assembly, collective exploration, coordinated motion, collective transport (partially), collective perception	open-source
		Swarm-bot		35 [149]		open-source
		Swarmoid		n.a.		n.a.
		ARGroHBots		15 [277]		n.a.
		Termes		5 [286]		open-source
		Symbion and Replicator		n.a.		n.a.
		PolyBot		32 [96]		n.a.
		M-Tran III		24 [218]		open-source
		ATRON		7 [46]		n.a.
		CONRO		8 [57]		n.a.
		Sambot		15 [407]		open-source
		Molecube		8 [429]		n.a.
Aerial		Puzzle Bot	UAV	9 [420]	pattern formation, self-assembly, self-reproduction	open-source
		Acoustic Swarm		5 [176]	collective exploration, coordinated motion, collective perception	open-source
		MAV		n.a.	n.a.	n.a.
		Distributed Flight Array		9 [279]	self-assembly, coordinated motion	n.a.
Aquatic	Environmental Monitoring	Crazyflie 2.1	UAV	49 [293]	aggregation, collective exploration, coordinated motion, collective localization, collective perception	open-source, commercial
		FINKen-III		n.a.		n.a.
		CoCoRo	UUV	41 [330]	aggregation, collective exploration, collective localization, task allocation	n.a.
		Monsun		n.a.		open-source
Outer Space	Space Exploration	CORATAM	USV	12 [94]		open-source
		microUSV		4 [146]		n.a.
		Swarmies	UGV	20 ¹	collective exploration, collective localization	n.a.
		Marsbee	UAV	3 [192]	collective exploration, coordinated motion, task allocation	n.a.

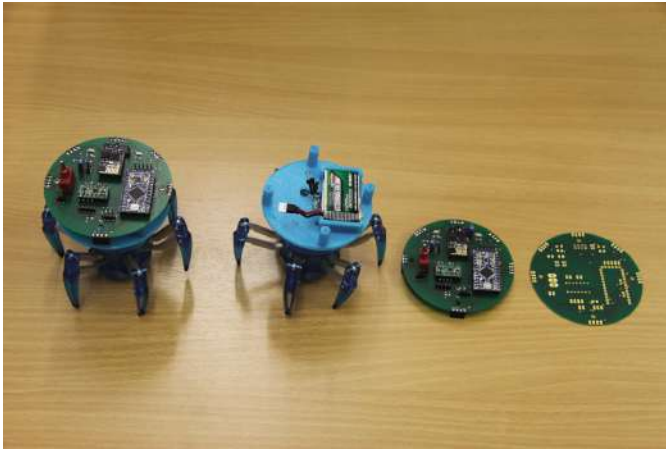


Figure 6.6: Spiderino robot (left) and its components. ↵

In the case of the Spiderino robot, the control system has been put atop an existing system that comes from a Hexbug Spider toy model (see Figure 6.6). The Hexbug Spider toy is a remote controlled plastic toy featuring six legs being controlled by two motors, one for moving the robot forward or backward and the other for turning the robot. A smart mechanical design enables the robot to move by lifting three legs at a time in the direction the robot's head is facing. The toy, being mass-produced, comes at a retail price between 15 and 30 €, making the system economically attractive even if the parts for the remote control are removed and not used in the robot.

To build your own Spiderino robot, follow these steps:

Electronics: The PCB file of Spiderino is available for download at the Spiderino website³⁵. The main board is a two-layer, circular board of 80 mm diameter. The top layer contains most of the components, including

- Socket for Arduino Pro Mini.
- Connector for an ESP8266 WiFi Module.
- Socket for motor driver (POLOLU Motor DRV-DRV8835).
- Three-way power switch.
- Three-pin configuration jumper.
- 5 resistors and 2 LEDs, both in 1206 SMD format.

³⁵Spiderino website: <https://spiderino.nes.aau.at/index.php/downloads/> [Online; accessed 26-March-2023].

Connecting the WiFi module is optional since the robot can also be programmed through a USB cable and is capable of operating without a WiFi connection.

The bottom layer of the PCB includes 6 connectors for sensor boards and connectors for the two DC motors.

The sensor board is a very small 10 by 10 mm 2-layer PCB that connects through four pins to the main board. Two pins are used for GND and VCC, while the other two pins contain a digital I/O and an analog input. The board can fit different sensors. The current sensor boards have an optical CNY70 distance sensor that points radially outward from the robot. Up to six sensor boards can be fit onto the robot, but it is also possible to have a configuration with fewer sensors.

Mechanical structure: In addition to the PCB, a Hexbug Spider toy robot and two 3D-printed adapters are required. The toy's main body has to be separated from its head containing the IR remote control interface since control will be implemented through the PCB. The main board is connected through screws to a 3D-printed adapter board that holds the battery and contains a cut-out for the motor cables. This part is glued to another conical adapter that fits the lower part of a Hexbug Spider toy robot, which contains the motors and provides a mechanism for coordinated leg actuation.

Programming: The robot's control program has to be uploaded to the Arduino Pro Mini. The Pro Mini can be programmed with Arduino IDE (available for Windows, Linux, and MacOS systems) in C or Atmel assembly language. The C-language API consists of functions for controlling the robots' motion and reading the sensors. Its implementation is documented in the technical documentation available for download at the Spiderino webpage[180].

6.5.2 A Wheeled Robot: Mona

Mona wheeled robot [17] is an open-source miniature robot that was mainly developed for long-term swarm robotics applications. It was first tested for a Perpetual Robot Swarm [24] system that aimed to develop a low-power robotic system with extremely long autonomy time, over weeks or months. Mona is an autonomous mobile robot with five proximity sensors, utilizes an AVR microcontroller as its main controller and can be programmed by popular Arduino IDE. The Mona robot has been commercialized³⁶ however it is still an open-source platform.

Two off-the-shelf DC motors with direct reduction gears were used as the main actuator. As shown in the architecture of the robot in Figure 6.7, a symmetrical differential driven configuration is used to control the robot's

³⁶Mona robot commercial website: <https://ice9robotics.co.uk/> [Online; accessed 26-March-2023].

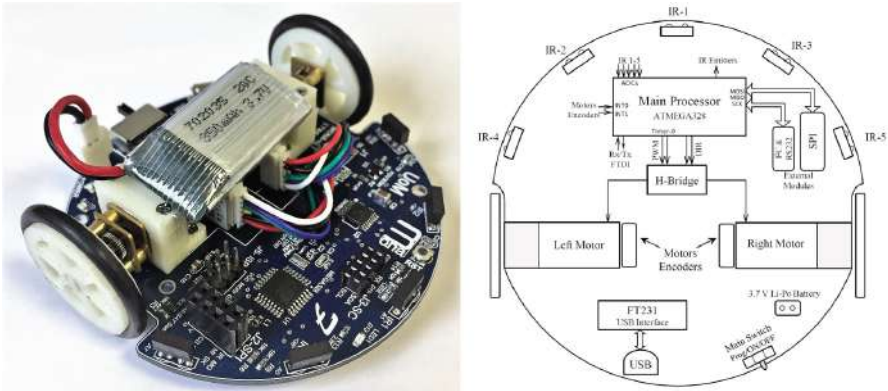


Figure 6.7: (left) Mona robot main platform. (right) The architecture of the basic platform [17]. ↵

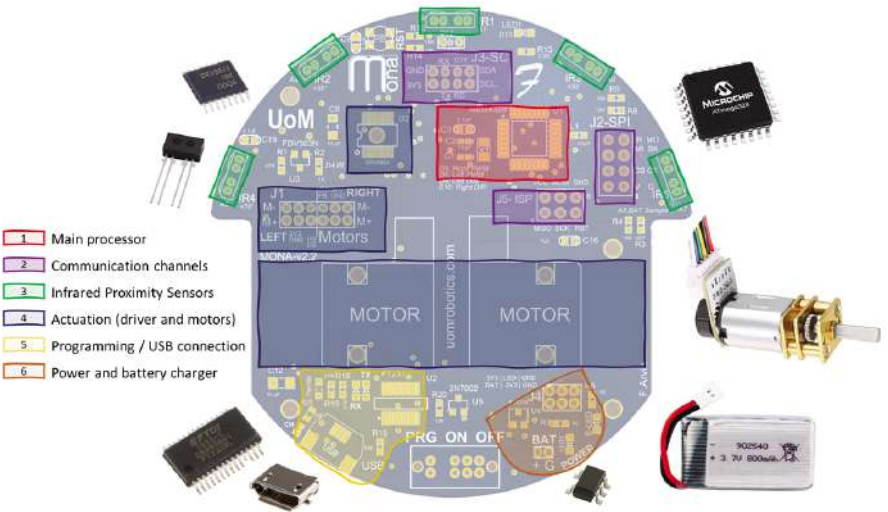


Figure 6.8: The Mona main-board and various sub-modules. ↵

motion with two motors. The robot’s wheelbase is 80 mm with 28 mm wheel diameters. The rotational speed of each motor is independently controlled through a pulse-width modulation channel of the controller, which is connected to a H-bridge motor driver.

To build your own Mona robot, one has to consider three crucial steps:

Electronics: The PCB file of the Mona is available online³⁷. It is a two-layer PCB and most of the components are on the top board. The main board has six functions (as shown in Figure 6.8) including: (1) The main controller that is an ATMEGA328, (2) communication connectors supporting RS232, serial peripheral interface, I²C (Inter-Integrated Circuit) and ISP, (3) IR proximity sensors, (4) motor driver and feedback link from motor encoders, (5) USB port and asynchronous serial data module used for programming the robot with Arduino, and (6) 3.7 V LiPo battery and the charging unit.

Mechanical structure: The main PCB also acts as the chassis of the robot, so motors, battery and all mechanical components sit on the main-board. This makes the assembly and controlling of the robot easier. In terms of mechanical structure, there are two parts—wheels and motor brackets—that must be 3D printed. The CAD design of the parts is also shared at the Mona GitHub.

Programming: The robot is compatible with most of the open-source programming platforms. Although the architecture of the robot enables programming Mona with any Arduino-based platforms via a USB cable, any programming language that is available for the AVR μ Cs, including C, C++, Java, Pascal, Basic, and Assembly can also be used for programming the robot. Hence, any software platform and IDE that support ISP can be used for programming the robot.

The latest version of the Mona robot was equipped with an ESP32 micro-controller providing additional functionality such as WiFi, Bluetooth and RGB LEDs. There are various available examples of robotic modules used for undergraduate engineering students. It includes a library (*Mona.h*), several examples for controlling motors and an autonomous obstacle avoidance scenario available at: <https://github.com/MonaRobot/Mona-Arduino>.

6.5.3 Another Wheeled Robot: Mechalino

The design of Mechalino is guided by the requirements for

- A low-cost robot (around 150 € per robot) so that a swarm of these robots is still affordable,
- Using wheels with good grip and stepper motors to have accurate movement control,
- Having a mostly cylindrical body in order to avoid getting stuck on edges,

³⁷Mona Test Github: <https://github.com/Farshad-Arvin/Mona-Platform/blob/master/Mona-Test.ino> [Online; accessed 26-March-2023].

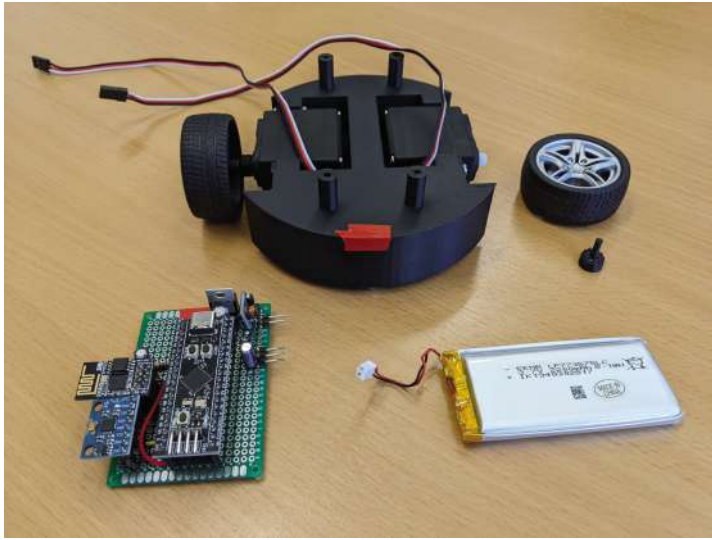


Figure 6.9: Mechalino components: Chassis, wheel, axle, mainboard, battery.
 ↙

- Providing an area on top for adding a marker that can be detected by an overhead camera.

While still reasonably priced, Mechalino is designed to support ROS and provide the processing power for running more elaborated algorithms and evaluations onboard. It is equipped with a STM32F4 Microcontroller Unit (MCU). The software and hardware design is available under an open-source license³⁸. For easy reproducibility of the hardware, the design aims to use standard components in combination with parts that are easily producible via 3D printing. Figure 6.9 shows the main components of a Mechalino robot. The assembled prototype is shown in Figure 6.10. The color markers on top allow for the accurate detection of the position and orientation of the robot from an overhead camera. The robot has a low center of gravity, facilitating precise movement. On the hand hand, the low ground clearance requires to operate the robot on a smooth surface. Mechalino is still under development and is being continuously improved.

To build your own Mechalino robot, follow these steps:

Electronics: The PCB file is available for download at the Mechalino webpage. The two-layer PCB fits the microcontroller, an ESP8266, a GY-521 gyroscope/acceleration sensor, two driving circuits for the left and the right motor and distance sensors.

³⁸Mechalino website: https://github.com/smartgrids-aau/Mechalino_Arena [Online; accessed 26-March-2023].



Figure 6.10: Assembled Mechalino with ArUco marker on top. ↵

Mechanical structure: The main body features a 3D-printed chassis with cut-outs for the motors and the battery. At the top, there are screw holes for attaching the main board, also allowing for a possible second-layer processing board. The wheels are pressureless standard rubber wheels from a model shop. A 3D-printed sleeve ensures a snug fit to the stepper motor axles.

Programming: The robot can be programmed in C++ and in ROS using MicroPython³⁹. A hardware abstraction layer needs to be uploaded onto the STM32F4, which is available at the project page. Programming is done through a cable, while the ESP8266 is utilized for communication at runtime.

³⁹MicroPython website: <https://micropython.org/> [Online; accessed 24-December-2024].

Chapter 7

Open Challenges and Outlook

As applications become more complex, there are specific challenges that need to be addressed. Traditional technical solutions involve creating a final product that includes numerous components and sub-components. Each of these components has a significant size and complexity of its own and must interact with other components to provide their services in a specified manner. Even minor changes to a component's services or interface could potentially disrupt the entire system's functionality. Existing examples have shown potential for using swarm-based approaches in engineering in order to address issues that come with the growing complexity of systems via a self-organizing approach. In addition, technical applications become more complex due to an increasing number of networked components being involved on both sides, the environment and the technical solution. The previous chapters have introduced several examples showing how swarm-based approaches can be successfully employed in such an environment. However, these examples might just mark the beginning of using swarm systems as a new paradigm for designing and operating systems. The complexity of applications in terms of functionality, components, and dependencies among these components will rise further, which means that we can expect the existing challenges to grow, as well in their importance. To face these challenges, new methods, techniques, and approaches need to be developed. By successfully solving these challenges, integrating swarm-based approaches to overcome existing technological boundaries will be a promising direction to explore and increase the efficiency of technical applications. The following sections are adapted and extended from Schranz et al. [333].

7.1 Challenges in Swarms of CPSs

As already discussed in Section 1.2, the modeling, design and control challenges for CPS swarms become even more difficult as the autonomy of individual components must be combined with explicit attention to interdependencies and coordination, interoperability, distributed control and emergence of behaviors. Swarm intelligence can bring potential advantages in terms of adaptability, robustness, and scalability to systems. However, several research challenges remain, especially when it comes to particular applications. For example, applying swarm intelligence to the physical domain, aka CPSs. In this domain, the limitations of embedded systems have to be considered, as well as an incomplete knowledge of the state of the physical world. Addressing these challenges not only helps to overcome current limitations but also paves the way for future applications of swarm systems in various fields.

7.1.1 Challenges Deriving CPS and Swarm Properties

Today's technical systems are usually predictable and controllable. For example, entering a target temperature in an oven causes the expected reaction, namely that the oven heats up to the set temperature and maintains this temperature. In swarm systems, these properties are often not directly given, since such a system is characterized by local rules, interactions and thus the emergence of the global system goal from the bottom up. A control input for a swarm system is, therefore neither easy to realize, nor is it guaranteed that the result of an input is easily predictable. In the design of CPS systems, the question of predictions, for instance, about the next system state, movement, or information, has a direct impact on the ability to give the system certain configurations, such as manipulating parameters, controlling individual movements, or assigning specific tasks. Thus, in addition to the issues of predictability and controllability, any parameter setting, even if only slightly adjusted, can significantly affect the behavior and dynamics of an intelligent swarm system. In particular, for a CPS swarm system, predicting such effects is difficult because modeling of the physical parts of the system is usually subject to uncertainty [333].

In recent years, there have been several research works addressing these topics, in particular related to network science [135] and the observability of complex systems [232]. Many systems are characterized by a number of parameters, numerical and structural, whose effects on the resulting dynamics in a complex system can often be extreme, leading to phase transitions [360, 76, 42], self-organized criticality [402] and deterministic chaotic behaviors [369, 423].

In CPS swarms, these parameter effects come from several sources that are typically hard to model, including environmental effects such as weather, earthquakes, the (partially) unpredictable behavior of humans, or any other random disturbance coming from the real world. The difficulty lies in creating

precise models for timely adaptation to exhibit the desired robustness to the dynamics of the environment [34]. Therefore, swarm-intelligent systems may instead find limited use in life-critical applications such as the control of an airplane—a clear definition of the possible behaviors of the system cannot be provided in a CPS swarm system.

Literature reports several works on the analysis of formal models for optimization or decision-making scenarios. They mostly use the Ant Colony Optimization (ACO) and the PSO models in combination with optimization problems and analyze stability and convergence properties. The analyzed scenarios often rely on constrained models, such as simplified algorithms or additional assumptions [65]. Examples include using a random dynamical system, for PSO [105], Lyapunov stability analysis [188], or deriving graphical parameters from dynamical systems theory [383]). In many practical applications, ACO and PSO models become too complex for complete theoretical analysis [333]. Typically, performance analysis is done via empirical procedures, possibly restricted on the assumption of defined theoretical properties [333]. Examples of such an approach involve using a specifically designed objective function for convergence analysis of a PSO approach [66], reviewing self-adaptive PSO algorithms related to stable points in parameter selection [154], parameter optimization studies [422], running numerical tests and results using test matrices associating multi-target tracking and measurement [44], and applying experimental analysis of how particular design choices affect the quality and the shape of the Pareto front approximations generated in multi-objective ACO [235].

In contrast to swarm intelligence systems in general, a formal model of CPS swarms to evaluate predictability and controllability is much more sensitive. In CPS swarms, we do not only consider the swarm itself, but also the different performance objectives, metrics, constraints, requirements, and disturbances coming from the real-world problem scenario. In addition to that, we have the cyber (behaviors and interactions) and physical (hardware-related) characteristics of each individual CPS. This means that each variation of the CPS, its behaviors, the scenario they operate impose a new set of configurations and thus, challenges in practice. This makes it difficult to abstract general properties and reuse results across different scenarios [333].

Nevertheless, with extensive mathematical abstractions it is possible to identify and analyze a basic set of swarm behaviors, including, foraging [58, 136, 95, 162], coverage [424, 338, 237, 69], aggregation and pattern formation [134, 13, 136], and cooperative tracking [424, 142, 340] (see Section 1.1.1 and Section 4.1 for more details); an extensive, and well-discussed list of swarm behaviors can be found in Brambilla et al. [45], Bayindir [31], and Schranz et al. [332].

7.1.2 Challenges Designing the Local Rules

The design of swarm intelligence often follows a bottom-up approach, where the desired global behavior is implicitly determined by defining local

interaction rules. This requires solving an inverse problem using heuristics to guide the design of the local rules. In swarm intelligence scenarios that are not tied to a physical scenario, for instance, generic optimization applications of swarm intelligence [301], the definition of heuristics that lead to a desired global behavior is easy because they are predetermined by the optimization goal.

However, in embodied swarm applications, such as CPS swarms, this task is much more complex due to the multi-layered, uncertain and hardly predictable structure of the environment. Therefore, most of the work in this direction is devoted to the automatic synthesis of behavioral and interaction rules for swarm robots. In Fehervari and Elmenreich [111], artificial evolutionary techniques (provided by FREVO [359]; see Section 4.5 for more details) are used to generate neural controllers for homogeneous robot soccer teams. In Gomes et al. [144] they apply novelty seeking to generate neural controllers for homogeneous robot swarms. Ferrante et al. [115] generate controllers for robots by combining the use of an evolutionary approach with a formal language. In Lopes et al. [234], supervisory control theory, a formal language based on a discrete-event representation of the system, is used to automatically generate controllers with built-in proofs and wider reusability in different robotic systems. Similarly, in Francesca et al. [124], the automatic design of robot swarm control software is achieved by generating a probabilistic finite state machine resulting from an optimization process that maximizes a task-specific objective function. Furthermore, in Tuci et al. [384], different evolutionary approaches are used to generate task assignment mechanisms that efficiently adapt robot behavior to the environment.

In general, these approaches, even if exemplified in a specific application case, can be extended to other domains. For example, adapting an algorithm from autonomous ground robots to autonomous aerial vehicles should be possible after considering additional constraints (aerial vehicles cannot easily bump into each other without damage) and degrees of freedom (height as a third dimension). All of these works, among others, mark valuable contributions to the automatic generation of rules that can be reused or expanded for the more general case of CPSs. However, it is important to acknowledge that these approaches have only been validated in relatively simple scenarios and have been applied exclusively to homogeneous swarms. As a result, these methods need to be better equipped to handle the complexities of heterogeneous swarms in CPSs.

Furthermore, these methods face challenges when dealing with large and complex search spaces, leading to what is commonly known as the dimensionality curse. Consequently, current automatic generation systems may not be feasible for systematic use when synthesizing complex behaviors required by CPS swarms. It is worth noting that a convergence between model-checking methods [64] for general CPSs and automatic rule synthesis and verification in physical swarms will be necessary to provide guarantees in both the design and control stages of CPS swarms. Such a combination would

ensure a systematic and reliable approach to address the challenges associated with CPS swarm systems.

7.1.3 Real-World Deployment Challenges

To date, most of the work implemented and tested on physical swarms has focused on swarm robotic research platforms. These works typically involve small, simple robots with basic sensors and limited data exchange capabilities. Although they exhibit interesting self-organizing behaviors, they serve mainly as test demonstrations and are difficult to scale for real-world applications. Their reliability, predictability, and efficiency need to be reevaluated when the same approach is to be applied to complex tasks in realistic environments. In contrast, contemporary technological advances have produced impressive individual CPS and robotic systems. Examples include self-driving cars developed by companies such as Uber and Google, remarkably stable quadrupedal robots such as Boston Dynamics' Spot, and advanced humanoid robots such as Boston Dynamics' Atlas, Hanson Robotics' Sophia, and Honda's Asimo. These systems are highly complex mechanically, algorithmically, and behaviorally. Compared to these cutting-edge examples, the tasks performed by current swarms of robots seem rudimentary. The question arises whether swarm robotics or specifically CPS swarms, can progress from being study objects to becoming systems producing real-world systems that can reliably perform useful tasks.

This transition may require a move away from overly simplified robot models and controls toward designs that achieve a better balance between simplicity and the ability to perform complex tasks effectively and reliably. It could entail transitioning from minimal resource usage and restricted information exchange to a more extensive utilization of sensor data and enhanced information sharing. Intel's Shooting Star [310] drone swarm provides an illustrative example in this regard. During the opening ceremony of the 2018 Winter Olympics, Intel orchestrated an impressive aerial show with 1218 drones. Although labeled as a swarm, the system is not really swarm intelligent; it operates as a partially distributed system with a central controller. Individual drones follow pre-computed trajectory scripts and rely on a precise external positioning system for navigation. The main idea is that to effectively perform real-world tasks, it might be essential to abandon overly restrictive assumptions and adopt a pragmatic, balanced approach by combining bottom-up swarm intelligence design with top-down methods whenever feasible.

Another significant challenge associated with the implementation of real swarm systems is the maintenance of such systems throughout their operational lifetime. As discussed in previous research [97], although they are swarm systems, complex technical systems that must operate reliably over a long period of time will require maintenance to ensure uninterrupted service. It remains an open question whether maintaining a swarm system with its unique characteristics will be more accessible or more challenging than traditionally

designed technical applications. In essence, maintaining a system composed of numerous autonomous components will be a challenging task. However, multiple components within the system can also provide opportunities for internal monitoring and troubleshooting. A relevant example is given in Christensen et al. [63], where failed robots are promptly detected and isolated from the swarm of robots, demonstrating the potential for addressing problems within the system itself.

7.1.4 How to Address the Challenges

In order to enhance the potential for formal analysis and promote the systematic reuse of results in swarm intelligence, it is advantageous to position swarm intelligence within the wider framework of complex systems and network science. This approach allows the adoption of sophisticated mathematical and modeling tools commonly used in the study of complex systems, which can help characterize properties such as evolution over time, stability, and structural/topological aspects. The focus on structural and topological properties has recently gained significant attention, as information flow plays a crucial role in self-organization and emergence [143, 79].

In addition, it is vital to identify the relationship between swarm intelligence and the fields of game theory and multi-agent systems [351]. Although this connection has yet to be adequately explored, exploiting insights from game theory and adopting mechanism design approaches can provide viable alternatives for generating interaction rules and analyzing the formal properties of swarm intelligence systems. Concepts such as equilibrium in game theory and interaction rule design through mechanism design [272, 41] present promising avenues for automatic rule generation. Given the current limitations of CPS swarms, it is expected that the study of physical swarms should align more closely with the multi-agent systems field while focusing on self-organizing behavior and scalability. The distinction between swarms and decentralized multi-agent systems is not always clearly defined. A recent survey of multi-robot systems [70] shows that the properties required for distributed control algorithms include locality of sensing and communications, scalability, security, contingency, and task-oriented considerations, similar to those addressed in research on swarm robotics [31]. This suggests that the differences between the two concepts are often semantic rather than substantive, with considerable overlap between them (for more details refer to the discussion in Section 1.1.3).

7.2 Future Trends and Directions

As pointed out in the introduction, there is a growing presence of complex dynamic systems that are increasingly collective and connected. These systems may be cyber or physical or, more generally, both, constituting a cyber-physical system. Given the nature and characteristics of these systems, swarm

intelligence will play an essential role in addressing the upcoming challenges. Given the vast solution possibilities and diverse challenges, new swarm models may be needed, and new fields of application will open up. In the following, we discuss where these new models might come from for inspiration and in what kinds of future applications swarm intelligence will prove to be a relevant tool. Finally, we outline general open research topics on CPS swarms.

7.2.1 Future Inspirations

Until today, most swarm intelligence models and algorithms have been inspired by social interactions within animal groups and animal societies [281]. Such observations are expected to remain a constant source of inspiration for new swarm intelligence models. However, we can also expand the range of potential inspiration models to include organic, inorganic, and social systems with similar fundamental properties. We can identify essential microscopic behaviors and local interaction rules by reducing these models to the key functional elements required in most swarm intelligence models. These microscopic interactions generate self-organizing behavior as an emergent feature at the macroscopic level.

One challenge is to build and extract abstract models from natural sources of inspiration. This includes exploring decentralized and time-efficient techniques from the specific scenarios studied in different research disciplines. Secondly, these models need to be used to build new models and address the problem complexity of real-world CPS applications.

The following sections illustrate specific areas that can provide new ideas for swarm intelligence model development. In some cases, a model has already been used as inspiration for swarm intelligence development. However, their potential to inspire swarm intelligence systems has yet to be fully exploited, so even an example known to have inspired swarm intelligence algorithms could be used to make new and potentially (more) disruptive contributions.

Biology

Animal societies have been the primary source of inspiration in swarm intelligence. Researchers and swarm intelligence practitioners have drawn inspiration primarily from cooperative animal societies, largely ignoring predator-prey systems and other conflicting scenarios [281]. However, to address cyber threats and the interaction among multiple, potentially competing CPSs, it is necessary to study non-collaborative systems. Research in population dynamics [174], game theory [351], and evolutionary game theory [322] has provided applicable models and ideas for the development of novel swarm intelligence systems. A promising approach involves the integration of autonomous robots into existing animal societies. Successful integration has been demonstrated with honeybees, fish, cockroaches, and cows [251, 326, 399]. This emerging field offers the potential to create a bio-hybrid system, or social cyborg, that combines the unique capabilities

of robots and animals in a mutually beneficial and symbiotic way. (*Programmable*) *Bacteria* serve as an inspiration for current and future research in various fields, including social intelligence. Their foraging [281] and collective decision-making abilities have allowed them to thrive and spread in challenging environments, making them a subject of interest for researchers seeking to develop new methods for solving complex problems [350]. In addition to their natural abilities, bacteria can also be programmed through genetic engineering techniques, providing opportunities for precise control and manipulation of their behavior [184]. For example, bacteria can be programmed to produce specific proteins or enzymes or respond to certain environmental signals. One exciting application of programmed bacteria is in the field of medicine. Recent research has shown that a swarm of magnetically-guided bacteria can be used to deliver drugs directly to tumors within the body, potentially improving the efficacy of cancer treatments [113]. This approach offers a promising alternative to traditional drug delivery methods, which can have harmful side effects and may not effectively target cancer cells.

Chemistry and Physics

Molecular networks have been studied across various fields [191], particularly for modeling complex diseases such as cancer and schizophrenia [147]. In these networks, nodes represent molecules like genes, RNA, and proteins; the edges represent their relationships. These network structures could inspire the design of complex interactions between agents in a swarm, as they exhibit outstanding properties such as scalability and resilience. For instance, the physiological interaction network of a single cell comprises thousands of chemical reactions that alter the concentration levels of hundreds or thousands of chemical components. Despite the enormous complexity and size of these networks, they exhibit quick and flexible dynamics while staying within precisely controlled bounds, thereby ensuring the survival of the organism. The properties of molecular and bacterial networks, characterized by their flexibility, robustness, scalability, and emergent complexity resulting from simple interactions, share remarkable similarities with the desired characteristics of swarm intelligence. Conducting laboratory experiments under controlled conditions provides an opportunity to study the characteristics of these networks. Because of the relatively small scale, a study can be conducted, for instance, on a few microliters of fluid in a cuvette or a small agar plate on a laboratory desk. Thus, these networks present ideal subjects for systematic exploration to provide valuable inspiration for developing future swarm intelligence algorithms.

Nanonetworks refer to a set of nanomachines that are interconnected and have various capabilities such as computation, data storage, sensing, and actuation. The research on nanonetworks has focused primarily on their non-traditional communication types, such as electromagnetic or molecular communication [3]. In particular, molecular communication between cells

using synthesis, transformation, emission, propagation, and reception of molecules has the potential to inspire swarm intelligence [4]. One of the main advantages of molecular communication is its ability to function in environments that are not conducive to electromagnetic signals, such as inside the human body. This makes it a promising avenue for designing swarm intelligence systems operating in similar environments, such as healthcare or environmental monitoring applications. Moreover, molecular communication offers a high degree of security and privacy due to the localized nature of the communication. Nanonetworks have already been successfully employed in biomedical applications such as drug delivery, in-vivo sensing, and monitoring glucose levels for diabetes patients. They have also been used in environmental applications such as pollution monitoring. Therefore, the study of nanonetworks and their communication mechanisms could serve as a rich source of inspiration for the design of future swarm intelligence algorithms. By leveraging the capabilities of nanonetworks, swarm intelligence systems could potentially achieve higher efficiency, scalability, and robustness in a wide range of applications.

Human Cognition

Humans also employ self-organizing strategies to interact with each other to solve various tasks [213, 215, 109], such as the collective patterns observed in crowd dynamics. Experimental studies have been conducted to identify corresponding behavioral rules, such as investigations of pedestrian behavior in single avoidance tasks [258]. In a study by Tavakoli et al. [377], they observed the coordination abilities of humans with limited perception in an environmentally distributed environment. This first experiment aimed to learn human-inspired behavior before formal strategies, potentially adaptable to CPS swarms, were developed. Surprisingly, compared to the multitude of algorithms derived from (eu)social animals, very little research has focused on extracting behaviors from human groups or societies to convert them into swarm intelligence algorithms [282]. This is unexpected since humans, being the creators of advanced and complex societies and cultures, are a readily available source of inspiration, easily accessible through natural language communication.

Interdisciplinary Approaches

The swarm-like concept of *active matter* consists of numerous individual agents following simple rules, resulting in collective behaviors and movements. Active matter combines representations from various disciplines, such as biology, physics, computer graphics, and robotics. A well-studied model in this domain is the self-propelled particle model proposed by Vicsek et al. [394] in 1995. This model has been extended to simulate various realistic systems, including self-replicating morphogenesis, in which swarms form structures that create new similar structures [328].

The researchers described collective motion in active solids and crystals, applying elasticity-based mechanisms to achieve self-organization [117, 116]. These models can be adapted to study the collective movement of CPS swarms, such as pattern formation or morphogenesis. They offer robustness to heterogeneity, which can be derived analytically and is particularly suitable for CPSs.

7.2.2 Promising Future Applications

Although the current implementation of swarm intelligence in real-world scenarios remains limited, we recognize the vast potential for its application in a wide range of practical situations. As Bonabeau and Meyer have pointed out, the possibilities for swarm intelligence application are limited only by our imagination [40]. This statement strongly supports our hypothesis, stated in the introduction, regarding the ubiquitous presence of swarm intelligence in meeting the challenges of an increasingly interconnected world.

Swarm intelligence has already been successful in various complex optimization scenarios in the form of algorithms such as ACO [84] and PSO [197], where swarm members encode individual solution candidates. However, physically embedded swarm algorithms have seen limited applications so far, with most engineering solutions following more traditional top-down or centralized approaches.

We expect that swarm intelligence will increasingly find applications in real-world scenarios where a top-down approach lacks sufficient information, a centralized approach is computationally infeasible, or real-time constraints impede the ability to find optimal solutions. These situations often arise in complex CPSs or scenarios without existing infrastructure, where the need is to provide ad-hoc working solutions without the ability to first establish an infrastructure framework (e.g., in disaster scenarios [78]). Furthermore, we foresee significant potential for swarm applications when transitioning from well-defined environments such as planned factories or warehouses to more complex environments such as traffic systems, urban environments, social networks, extreme/hostile environments, or uncharted territories.

Looking ahead, we consider the next frontier for swarm intelligence in CPS swarms. In this regard, we highlight specific application scenarios that are currently (partially) unattainable but have the potential to take full advantage of swarm characteristics. Much research already exists, many of which await system integration to meet future application requirements. In the following examples, we describe the intended application of CPS swarms, outline their swarm characteristics, and identify areas of research that require further investigation to achieve the desired applications, where the term “CPS swarm” can be understood as an umbrella term.

Autonomous Driving and Smart Traffic

Autonomous driving is a prominent and much discussed topic, presenting various computational challenges in dynamic environments requiring real-time processing capabilities. This is where the concept of swarm intelligence becomes relevant: Thousands of cars, each with different levels of autonomy, as well as the road infrastructure, collaborate to find common solutions. Applying a swarm model to autonomous cars and infrastructure in an intelligent traffic management system offers several benefits, including reduced traffic congestion, coordinated creation of emergency lanes, improved traffic flow, and reduced carbon emissions. One potential application is swarm or fleet navigation, incorporating information from other swarm members, infrastructure, and the dynamic environment.

Modeling networked autonomous cars and smart infrastructure as a swarm is both feasible and beneficial for several reasons:

1. Each CPS can retrieve and evaluate information locally, following local rules.
2. Local information can be exchanged between CPSs to form a global or swarm-level perspective.
3. The infrastructure is relatively known in advance or subject to slight dynamics, such as accidents or construction sites.

A central traffic management system would have difficulty coordinating and managing each CPS participating in the traffic. At the same time, individual CPSs may cooperate locally, potentially leading to emergent behaviors.

While transportation logistics already makes extensive use of ant- and bee-inspired algorithms for route optimization [255, 421, 306, 426, 346], the concept of smart traffic remains relatively unexplored. However, the media frequently talks about swarms of autonomous cars and smart infrastructure [126, 339, 168], indicating the growing interest in such applications. However, implementing these swarms requires careful consideration, especially in terms of management. Questions arise, such as: What motivates CPSs to form a swarm? What common features, properties, or sensor data should CPSs share? How do CPSs join or leave a swarm? What decision-making processes should prioritize local driver goals, swarm goals, or global smart traffic goals? What is the optimal number of CPSs needed to form and benefit from a swarm? Integrating the data obtained from the infrastructure into the autonomous car swarm could further improve the decision-making process to achieve the above goals.

The technology to realize CPS swarms in smart traffic partially exists today. Local retrieval and evaluation of sensor data in cars is already common practice, and efforts are underway to develop standardized communication protocols for vehicle-to-vehicle and vehicle-to-infrastructure communication, collectively called Vehicle to Everything (V2X) [59]. V2X involves communication between cars and smart infrastructure integrating

physical elements like streets, cameras, traffic lights with digital infrastructure like sensors and communication networks. Communication technologies such as 5G [12] and Ultra-Wideband [152] offer high levels of security, low latency, and high data rates, enabling self-location and real-time data exchange between CPSs. These advances enable a swarm of autonomous cars to share and react to real-time local information, adapting to dynamic situations. For example, in Ulbrich et al. [388], the authors describe a swarm behavior module that selects and summarizes sensor data and performs trajectory planning for an autonomous car. In Li et al. [227], they develop swarm-intelligence-based local rules to enable the efficient operation of interconnected autonomous cars on arterials.

Honda's SAFE SWARMTM concept is an example of improving traffic by using information from vehicles in front, including onboard sensors, systems, and V2X [256] communication. Implementing an efficient and effective intelligent traffic system would likely require a complete redesign of the existing architecture for real-time data collection, processing, and analysis. In addition to the swarm system, a central Traffic Management System (TMS) could be implemented to fulfill several supporting roles. First, the TMS would function as a data submission system, ensuring that local event information is available to all other cars so they can plan their routes accordingly. Second, the TMS would provide an overview of global traffic situations by integrating and merging information about individual road swarms. This would provide a global understanding of traffic conditions and facilitate more effective decision-making. The data exchange process between cars and the TMS can be further enhanced by incorporating information from the intelligent road infrastructure. By integrating data from sensors embedded in the infrastructure, such as traffic lights or cameras, the overall system can gather more complete and accurate information about road conditions. In summary, the development of an intelligent traffic system would involve not only the implementation of a swarm system for autonomous cars but also the implementation of a central TMS. This would require a new architecture for real-time data collection, processing, and analysis. By incorporating information from both the vehicles and the smart road infrastructure, the system can achieve higher levels of efficiency and effectiveness in managing traffic flows.

Emergency Response

The presence of professional first responder teams in emergency situations is critical to saving lives and restoring essential services. Traditional first responder teams consist of police, military, firefighters, and rescue units from civil protection agencies. However, the use of CPSs swarms can greatly enhance and facilitate emergency response activities. These CPS swarms, including different types of robots such as all-terrain or flying robots [78, 325], can work collaboratively with human teams, creating a heterogeneous, multi-human CPS swarm. These mixed teams can effectively handle accidents at

industrial or power plants, as well as environmental disasters such as wildfires, floods, storms and earthquakes. CPS agents within the swarm have the potential to navigate destroyed or inhospitable environments more efficiently and safely than humans. They can collect multi-sensor data, coordinate with human team members, focus on areas of interest like potential survivors, remove obstacles, restore structures and save lives. In essence, a CPS swarm can serve as a dynamic and distributed augmented sensor-actuator system, providing invaluable support to human first responders in effectively and efficiently conducting their missions.

Joint teaming with CPS swarms is feasible under several conditions: (i) Each individual swarm member can operate under local rules and is independent of the fixed infrastructure. (ii) The infrastructure is often disrupted, with blocked or unstable paths, making it a dynamic environment. (iii) Individual information can be exchanged between swarm members, other swarms and local control stations. (iv) A dynamic ad-hoc communication infrastructure is established to replace the pre-existing infrastructure that may no longer be available. This allows for intra- and inter-swarm communication, as well as communication between the swarm and control centers. (v) High-level decision making can be done in the control centers, balancing local autonomy with system-level decision making. (vi) Swarm members and human teammates can interact and exchange information transparently at the local level. (vii) Cooperative and collaborative behaviors are encouraged at the individual level to enable effective behaviors to emerge at the global level.

To effectively deploy mixed teams (heterogeneous swarms) in real-world scenarios, substantial progress is still needed, particularly in autonomous decision-making, data exchange and fusion, and operation in complex and unexplored environments. Research efforts are underway to address these challenges in heterogeneous swarms, such as those involving drones and ground rovers working together to provide first aid or guide victims to emergency exits. For example, Saez-Pons et al. [319] developed an autonomous robot swarm for SAR applications. They used social potential fields to generate formations and navigate while maintaining them. Although several other projects in this area [80, 238, 372], none have yet produced applicable results for real-world missions. The design and deployment of a heterogeneous swarm for such missions involve various open questions, including determining the number, type, and initial location of robots [71].

In the context of human interaction in mixed teams, it is essential to design intuitive and seamless methods for multimodal dialogue and data exchange between humans and CPS. This requires developing user-friendly interfaces and controls [148, 314, 216, 92]. Infrastructure, including communication networks, is often destroyed in emergency response applications, such as post-earthquake scenarios. Therefore, new relaying concepts are needed to enable the CPS swarm to create its own communication network. For example, Hauert et al. [156] developed a swarm of autonomous micro aerial vehicles capable of deploying and managing an ad-hoc Wi-Fi network. Furthermore, the highly coordinated and collaborative nature of the swarm members is

crucial in the disrupted environment. This coordination can be achieved by exchanging status updates or, if bandwidth permits, by sharing complete images and detected points of interest. It is important to develop compact image processing algorithms that align with local processing capabilities.

Environmental Monitoring

Monitoring Earth's threatened ecosystems is an emerging application for large-scale autonomous CPS swarms. These swarms can function as active sensor networks that adapt based on previously collected and assessed habitat data. This enables monitoring points of interest with greater accuracy, achieved by increasing the density of CPSs or improving the frequency of measurements at specific locations.

Supporting environmental monitoring through CPS swarms is feasible due to several factors: (i) Each individual swarm member operates under local rules, facilitating autonomous behavior. (ii) Information exchange occurs between CPSs to determine new collective behaviors. (iii) The natural environment being monitored is generally unknown and subject to dynamic and unpredictable events. (iv) Environment monitoring is intended to avoid disruption from external intruders, emphasizing the need for a high degree of autonomy. (v) Cooperation among individual swarm members leads to emergent behavior, enabling achieving monitoring objectives.

Swarms are well suited to operate in unexpected environments and can compensate for the loss or failure of swarm members. Therefore, this technology is an ideal candidate for long-term autonomous operation [83] during unpredictable events in unknown environments. For environmental monitoring, lightweight and efficient communication protocols between CPS [331] are especially appropriate. Excessive communication can disrupt the natural communication network of the ecosystem. Minimizing CPS communication and actions preserves the undisturbed nature of the environment, making it easier to analyze authentic natural behavior and processes. Therefore, adopting a principle such as "act only when necessary, but do so intelligently and gently" aligns with the behavioral optimization found in natural organisms. In addition, light and efficient communication protocols contribute to energy conservation. Since these swarms are typically deployed for months or even years to monitor ecosystems, smart energy consumption must align with the observation time.

A CPS swarm designed for environmental monitoring can consist of actively moving agents, allowing them to perform a crucial task at the termination of the monitoring process: Self-removal of the swarm from ecosystems in an environmentally friendly and sustainable manner, ensuring no harm is done. Several research projects are known to have successfully used real robots for environmental monitoring, including subCULTron [380], CORATAM [94], and CoCoRo [330].

Electric Energy Grids

Electric energy grids are highly complex interconnected systems and, therefore, prime candidates for swarm-based solutions. Venayagamoorthy [393] identifies several application areas where computational intelligence can be used in electric power grids, such as algorithms for energy and power flow management, voltage and reactive power control, dynamic load forecasting, and vehicle-to-grid integration. Although swarm algorithms are not explicitly mentioned as examples, many of these coordination tasks might benefit from swarm intelligence approaches. In addition to using swarm-based optimization algorithms such as PSO, there are approaches that directly model a portion of the electric power grid as a swarm. For example, Elmenreich et al. [102] present an example of coordinating homes in a neighborhood using a swarm-like approach to demand management. This approach uses geographic proximity to ensure fairness among consumers in the neighborhood. Steber et al. [364] describe a virtual mass storage system consisting of distributed battery energy storage units installed in homes with rooftop photovoltaic systems. Another example is the SmartGRID concept presented in Huang et al. [173], which proposes a decentralized and interoperable grid scheduling framework using a swarm-intelligent approach. Ramachandran et al. [304] proposes a hybrid swarm-immune-based auction system to coordinate generation and consumption in a smart microgrid. Black electrification using microgrids is another application that integrates small off-grid systems into an interconnected microgrid, as discussed in Kirchhoff et al. [205]. This approach, known as black-based electrification, is particularly relevant for rural, underdeveloped areas [220, 260].

The consideration of using CPS swarms to model electric power networks is compelling for several reasons: (i) The increasing use of distributed renewable energy sources is leading to more decentralized systems. (ii) Renewable energy sources such as solar PV and wind power cannot easily adjust their output to meet higher demand, making centralized control challenging. (iii) Electrification of transportation leads to increased electricity consumption by numerous individual consumers. These factors require a transition from centralized control of a few manageable power plants to a self-organized, swarm-like system in which generators, storage, and consumers interconnect in a distributed manner.

Practical deployment of CPS swarm models in real power grids requires significant research progress. Given the critical nature of electric power grids as essential infrastructure, validation and testing of such systems prior to deployment is a frequent requirement and an open question with respect to the current state of the art. Without validation and testing before deployment, implementing new control strategies, even in a small part of the system, can cause oscillations and affect the stability of the overall system [141]. Addressing this challenge may involve simulation at a granularity and accuracy so that the reality gap is negligible [257] or the application of CPS swarm models within smart microgrids [357]. Smart microgrids provide

an environment large enough to benefit from a swarm-based approach while being isolated from the national grid, reducing potential risks.

Space Missions

Space missions offer tempting opportunities to use swarm-based CPS approaches. One compelling application is the inspection of communication satellites, where swarms can be used to assess hull damage and ensure optimal performance. In addition, coordinated swarms can be instrumental in addressing the space debris issue by performing disposal operations. Another interesting concept involves using a cluster of reflective spheres to measure the gravitational force of asteroids in our solar system. These missions involve exploring water, raw materials, and potential life on planets and exoplanets, which serve as potential precursors to human colonization efforts.

Key features required for these missions include low-cost performance and fault tolerance, which can be achieved by using large CPS swarms that operate autonomously in unknown environments [368]. Establishing reliable communication channels to relay the data collected by the swarm back to Earth is crucial. For example, the Marsbees concept [219] envisages using the Mars rover as a base and recharging station for these autonomous CPS swarms. The Japan Aerospace Exploration Agency is taking this idea further by envisaging swarms of autonomous machines to prepare the ground, excavate and build facilities for astronauts on the Moon and Mars [419]. The first real-world tests were related to networking and communication in different spheres including Nodes [51], Proba-3 [291], and KickSat [204].

The feasibility of using CPS swarms for such visionary applications stems from several factors: (i) Each individual CPS operates based on local rules, which allows for autonomous behavior. (ii) Local information is exchanged between the CPS and stations on Earth, potentially using relay stations in planetary orbits. (iii) These deployment scenarios involve dynamic and unknown environments with no prior knowledge of the infrastructure. (iv) Centralized control is impractical due to the significant time delays between swarm operation and human intervention. (v) Successful achievement of mission objectives requires effective cooperation between individual swarm members.

Currently, the above missions still rely heavily on costly ground control for their operations. However, the ultimate vision is establishing autonomous satellite formations, as human intervention causes communication delays and associated costs. Formations are closely linked to the motion of spacecraft, which is influenced by factors such as gravity, solar radiation pressure, and atmospheric drag, causing deviations from the motion patterns observed on Earth [77]. The development of motion models becomes crucial for the guidance, navigation, and control of these formations. When direct exploration of planetary surfaces becomes feasible, autonomous swarms can be used. However, challenges remain related to transportation to the planet, launch procedures, navigation, ground surveying, and data retrieval. Factors such as

planet-specific temperatures, ground conditions, and gravity must be carefully considered during these operations.

Medical Applications

The field of medicine has shown a growing interest in using swarms to solve complex problems, such as cancer treatment. In healthcare, nanoparticles are important because they can leak out of blood vessels and go to their target sites. Nanoparticles play a key role in healthcare as they can penetrate blood vessels and target specific sites in the body. Although these particles are too small to be individually programmed, they can be prepared for swarm-based applications by changes to their coating, charge, or size. Suppose a group of nanoparticles exhibits swarming behavior. In that case, they have the potential to navigate to target cancer cells, carry a healing coating, be activated by external stimuli, and effectively destroy target tissues. Unlike other drugs that are dispersed throughout tissues, a swarm of nanoparticles has greater intelligence and can target diseased tissue more precisely.

Several factors support the use of nanoparticle swarms in medical applications: (i) Each individual nanoparticle operates based on local rules, which allows for autonomous behavior. (ii) Local information is exchanged between nanoparticles and possibly with an external monitoring station outside the body. (iii) In such scenarios, there is no existing infrastructure. (iv) The environment is a human or animal body, which is partly unknown and highly dynamic. (v) Achieving the desired goals requires cooperation between individual nanoparticles, leading to emergent behavior.

However, some fundamental questions remain unanswered when it comes to designing a swarm of nanoparticles for cancer therapy. For example, how to establish communication between individual swarm members, between swarm members and the environment, and between the swarm and the outside world? In addition, the implementation of swarm characteristics related to complexity, local intelligence, and local processing poses challenges. The swarm system must be simple, reconfigurable, cost-effective, scalable, and verifiable. Addressing these issues could provide insights into implementing more complex tasks such as optimization, computation, decision-making, construction, self-assembly, and collective movement within the swarm.

This vision of cancer treatment based on nanoparticle swarms is still in its early stages. However, it raises the question of whether other medical applications could be realized by injecting the swarm into the human body. Further research is needed to fully understand the possibilities and implications of this exciting medical frontier.

Human Networks

Human organizations and social networks, powered by Web 2.0 (defined as the transition from static websites to dynamic user-generated content and social media), represent collective systems. The swarm-like nature of social

networks becomes apparent when a large number of individuals interact and communicate locally, shaping global behavioral patterns such as online trends and the popularization of products or ideas.

In this context, swarm intelligence methodologies can be applied to study such systems from a swarm point of view. Human networks communicating over the Internet come with the advantage to provide vast amount of available online data sources. Analytical prediction and evaluation through simulation can help to predict and potentially control the behavior of other complex systems. Furthermore, swarm intelligence could find a direct practical application within human networks in two main ways: As an integral part of the network itself and as a physical service that can be used, traded, and negotiated by the network. We already use different types of digital assistants like Alexa [299], Siri [16], Cortana [166]) in our daily lives, which can be seen as some of the most advanced examples of CPSs in the real world. They communicate with people via text or speech, have full connectivity, are often mobile, perform computations to facilitate precise interactions with people, and exhibit proactive behavior such as suggesting or reminding tasks. With continuous advances, these digital assistants are becoming increasingly autonomous and capable of decision-making, adaptation, and lifelong learning. In the future, we can envisage the delegation of many tasks to these digital partners, allowing them to work on our behalf 24/7. When we reach this stage, the digital ‘society of us’ will parallel our physical society, among other things, performing information searches, making connections, and engaging in commerce.

The CPS swarm approach provides a framework for managing large-scale systems consisting of billions of agents. This approach offers several advantages: (i) Agents in a swarm act autonomously and are guided by a set of behavioral rules that reflect the personality traits and preferences of their human counterparts. (ii) Agents cooperate locally based on the concepts of a dynamic social neighborhood, while exchanging and sharing global information. (iii) The environments in which agents operate, including the emerging Web 3.0 and the physical world, which are highly dynamic, difficult to predict, and inherently parallel and distributed. (iv) Digital assistants, a form of CPS swarm, are designed to provide services based on gathered information. In a swarm of digital assistants, there is a constant exchange of information, searching, negotiation and trading between members. (v) Each agent in the swarm is inherently selfish and represents the preferences of its human double. However, with appropriate rules of interaction, socially aware patterns can emerge globally.

There are many practical applications for CPS swarms of digital assistants. For example, with appropriate consent and privacy protection, the continuous exchange of information between digital assistants can enable the tracking of physical locations. This capability could play a key role in scenarios such as epidemic tracking, where potential exposures can be identified and containment measures automatically implemented. In emergency situations, swarm assistants can take advantage of real-time information sharing to guide

the safe escape of individuals in a crowd, allowing for the emergence of a collective escape plan. Another possible application is coming from the circulation of fake news on social networks that presents a serious problem. Cooperation between different web miners working for different entities is often limited, which hinders the collection of comprehensive information and effective inference. Using a collaborative CPS swarm, a large amount of information can be collected in parallel, shared among members, and used for coordinated analysis and inference, allowing for more effective detection of fake news. Looking to the future, CPS swarms of digital assistants could work together to find, negotiate and trade with other CPS swarms operating directly in the physical world. This multi-level interaction between CPS swarms opens up new possibilities for service provision and the resolution of complex tasks. In addition, sensing as a service is a growing business approach in IoT, where users pay for data collected by specific sensors. This concept can be extended to swarms of CPSs, allowing them to offer sensing and actuation services in different locations. Members of the digital swarm can then find and negotiate the use and cost of available CPS resources for specific tasks, sharing experiences and collaborating with each other. This creates a complex society of CPS swarms interacting at multiple levels. Blockchain technology, known for its decentralized and distributed nature, will play a fundamental role in enabling secure transactions and building trust between swarm members.

In summary, CPS swarm approaches provide a powerful framework for managing large-scale systems, and digital assistants serve as conspicuous examples of CPS swarms. The potential applications of CPS swarms are diverse, ranging from tracking epidemics and responding to emergencies, to combating fake news and creating complex societies of interacting swarms.

7.2.3 Research Challenges

As swarm intelligence is still a relatively young topic, several crucial open research questions remain. In the previous subsections, we discussed domain-specific topics related to specific innovative applications. However, it is essential to note that many other application-specific research questions are not mentioned here. These issues are highly dependent on the specific context of the application. Additionally, there are general themes in swarm systems that are independent of the domain or application. We will use the example of CPS swarms to illustrate these themes. A swarm of CPSs faces several requirements that need to be addressed. These include designing, programming, and implementing highly distributed and connected digital technologies embedded in a wide range of devices. In addition, designing increasingly autonomous physical systems with diverse dynamics is challenging while meeting several critical constraints. In addition, addressing systems of systems with a high degree of autonomy is key to ensure scalability, adaptability, robustness, complexity management, safety, and security, and establishing trust between people and swarms. In order to provide an overview of open research topics in swarm systems, we group general topics into the

following categories. It is important to note that these categories address only a subset of the general topics, as a discussion of all the underlying details would be beyond the scope of this book.

Modeling

Creating models from natural or other sources of inspiration is an important topic in the research community. As mentioned in Section 7.2.1, many untapped sources of inspiration can still be explored. This research area also includes fine-tuning application-specific parameters for existing algorithms. In addition, algorithmic topics cover scalability, degree of control, degree of convergence, and the delicate balance between exploitation and exploration [418]. Furthermore, combining different sources of inspiration can be used to solve complex modern problems with self-organizing capabilities. The key idea is to adapt the application-specific swarm intelligence algorithms to fit the underlying problem definition.

Design

As swarms of CPS continue to proliferate in different application contexts and experience increasing acceptance, the challenge of designing systems that can efficiently achieve predefined goals while remaining flexible, reliable, and adaptable to changing conditions is becoming increasingly daunting.

The design of the next generation CPS must address multiple broad challenges, as described by Isaac et al. [67]. These challenges include: (i) Integration of complex, heterogeneous large systems, (ii) Interaction between people and systems, (iii) Dealing with uncertainty, (iv) Measuring and validating system performance, (v) Enriching systems with learning capabilities and (vi) System design.

From a fundamental point of view, individual theories provide formal descriptions of different aspects of CPS design, covering physical, technical, organizational, and human-system interaction aspects. However, these disciplines are not fully integrated into a single system theory. In other words, while methodologies, representations, and tools exist to address specific aspects of CPS design, there is still an open challenge in providing support for the complete design life cycle. In addition, a formal design methodology must include a process that iteratively revisits and refines micro-level behavior and bridges the gap between local and global behavior. Although Brambilla et al. [45] describe behavior-based design and automated design methods for CPS swarms, there is currently no implemented toolchain that can map these design processes. One possible approach to address this gap is the CPSwarm workbench [25], which integrates several tools to guide CPS swarm designers through the entire lifecycle, covering the design, optimization, simulation, and deployment phases.

Validation and Verification

Verification and validation in CPS swarms cover a wide range of systems and systems of systems, including hardware, software, information, processes, personnel, and facilities. It is crucial to explore robust methodologies for validating swarm behavior in CPSs. Whenever possible, it is important to define detailed standardized KPIs, scenarios, test areas, and benchmarks to ensure the reusability of measurement methodologies. The results obtained from verification and validation activities should feed back into the design specifications, model definitions, and validation.

Human in the Loop

Enabling human interactions with swarm systems [269, 210] offers many advantages, as CPSs can leverage humans' cognitive and sensory-motor capabilities. In contrast, humans can use CPSs in closed-loop interactions as augmented external sensor-actuator systems. Nevertheless, the design and implementation of such mixed-initiative systems [182] present scientific and technological challenges. These challenges encompass various aspects, including the physical performance of two-way interactions (between single CPSs or swarms of CPSs and humans), the presentation of the complex state of the distributed CPS to humans, and the dissemination of information or commands from humans to humans and CPS swarms. Considerable research has been devoted to the development of bi-directional interaction and dialogue interfaces and modalities, exploring the use of various multimodal interfaces [148, 49, 270, 185] for proximal interaction with swarms.

However, allowing direct interaction of human operators within a swarm system introduces new potential risks. The presence of humans can negatively impact the security, safety, stability, and reliability of the system response if they act irresponsibly or maliciously. These aspects have not received much attention to date but will become major concerns in the near future, especially with respect to security and privacy at all levels.

Bibliography

- [1] Russ Abbott. Emergence explained: Abstractions: Getting epiphenomena to do real work. *Complexity*, 12(1):13–26, October 2006.
- [2] Faisul Arif Ahmad, Abd Rahman Ramli, Khairulmizam Samsudin, and Shaiful Jahari Hashim. Optimization of power utilization in multimobile robot foraging behavior inspired by honeybees system. *The Scientific World Journal*, 2014, 2014.
- [3] Ian F. Akyildiz, Josep M. Jornet, and Massimiliano Pierobon. Nanonetworks: A new frontier in communications. *Communications of the ACM*, 54(11):84–89, 2011.
- [4] Ian F. Akyildiz, Massimiliano Pierobon, Sasitharan Balasubramaniam, and Yevgeni Koucheryavy. The internet of bio-nano things. *IEEE Communications Magazine*, 53(3):32–40, 2015.
- [5] Sergey Alatartsev, Vera Mersheeva, Marcus Augustine, and Frank Ortmeier. On optimizing a sequence of robotic tasks. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 217–223, 2013.
- [6] R. McNeill Alexander. Gaits of mammals and turtles. *Journal of the Robotics Society of Japan*, 11(3):314–319, 1993.
- [7] Antonio Luca Alfeo, Eduardo Castello Ferrer, Yago Lizarribar Carrillo, Arnaud Grignard, Luis Alonso Pastor, Dylan T. Sleeper, Mario G.C.A. Cimino, Bruno Lepri, Gigliola Vaglini, Kent Larson, Marco Dorigo, and Alex Sandy Pentland. Urban swarms: A new approach for autonomous waste management. *IEEE International Conference on Robotics and Automation*, 2019-May:4233–4240, 2019.
- [8] Muhanad H Mohammed Alkilabi, Aparajit Narayan, and Elio Tuci. Cooperative Object Transport With a Swarm of e-puck Robots: Robustness and Scalability of Evolved Collective Strategies. *Swarm Intelligence*, 11(3-4):185–209, 2017.
- [9] Michael Allwright, Navneet Bhalla, Haitham El-faham, Anthony Antoun, Carlo Pinciroli, and Marco Dorigo. Srocs: Leveraging stigmergy

- on a multi-robot construction platform for unknown environments. In *Swarm Intelligence: 9th International Conference, ANTS 2014, Brussels, Belgium, September 10-12, 2014. Proceedings 9*, pages 158–169. Springer, 2014.
- [10] Lee Altenberg. Advances in genetic programming. In Kenneth E. Kinneer, Jr., editor, *The Evolution of Evolvability in Genetic Programming*, chapter 3, pages 47–74. MIT Press, Cambridge, MA, USA, 1994.
- [11] Arash Sadeghi Amjadi, Mohsen Raoufi, Ali Emre Turgut, George Broughton, Tomáš Krajník, and Farshad Arvin. Cooperative pollution source exploration and cleanup with a bio-inspired swarm robot aggregation. In *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 469–481. Springer, 2020.
- [12] Jeffrey G Andrews, Stefano Buzzi, Wan Choi, Stephen V Hanly, Angel Lozano, Anthony CK Soong, and Jianzhong Charlie Zhang. What will 5g be? *IEEE Journal on Selected Areas in Communications*, 32(6):1065–1082, 2014.
- [13] Gianluca Antonelli, Filippo Arrichiello, and Stefano Chiaverini. Flocking for multi-robot systems via the null-space-based behavioral control. *Swarm Intelligence*, 4(1):1–37, 2009.
- [14] Miguel Aranda, Gonzalo López-Nicolás, Carlos Sagüés, and Youcef Mezouar. Formation control of mobile robots using multiple aerial cameras. *IEEE Transactions on Robotics*, 31(4):1064–1071, 2015.
- [15] Vincent Arnould, Laurent Balmelli, Ian Bailey, James Baker, Cory Bialowas, Conrad Bock, Carolyn Boettcher, Roger Burkhart, Murray Cantor, Bruce Douglass, Harald Eisenmann, Anders Ek, Brenda Ellis, Marilyn Escue, Sanford Friedenthal, Eran Gery, Hal Hamilton, Dwayne Hardy, James Hummel, Cris Kobryn, Michael Latta, John Low, Robert Long, Kumar Marimuthu, Alan Moore, Véronique Normand, Salah Obeid, Eldad Palachi, David Price, Bran Selic, Chris Sibbald, Joseph Skipper, Rick Steiner, Robert Thompson, Jim U’Ren, Tim Weilkens, Thomas Weigert, and Brian Willard. Systems modeling language (SysML) version 1.5. Standard, Object Management Group (OMG), May 2017.
- [16] Jacob Aron. How innovative is Apple’s new voice assistant, Siri? *New Scientist*, 212(2836):24, 2011.
- [17] Farshad Arvin, Jose Espinosa, Benjamin Bird, Andrew West, Simon Watson, and Barry Lennox. Mona: An Affordable Open-source Mobile Robot for Education and Research. *Journal of Intelligent & Robotic Systems*, 94(3-4):761–775, 2019.

- [18] Farshad Arvin, Tomáš Krajník, Ali Emre Turgut, and Shigang Yue. COSΦ: artificial pheromone system for robotic swarms research. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 407–412, 2015.
- [19] Farshad Arvin, John Murray, Chun Zhang, and Shigang Yue. Colias: An Autonomous Micro Robot for Swarm Robotic Applications. *International Journal of Advanced Robotic Systems*, 11(7):113, 2014.
- [20] Farshad Arvin, Khairulmizam Samsudin, and Abdul Rahman Ramli. Development of a Miniature Robot for Swarm Robotic Application. *International Journal of Computer and Electrical Engineering*, 1(4):436, 2009.
- [21] Farshad Arvin, Khairulmizam Samsudin, Abdul Rahman Ramli, and Masoud Bekravi. Imitation of Honeybee Aggregation with Collective Behavior of Swarm Robots. *International Journal of Computational Intelligence Systems*, 4(4):739–748, 2011.
- [22] Farshad Arvin, Ali Emre Turgut, Farhad Bazyari, Kutluk Bilge Arikan, Nicola Bellotto, and Shigang Yue. Cue-based aggregation with a mobile robot swarm: A novel fuzzy-based method. *Adaptive Behavior*, 22(3):189–206, 2014.
- [23] Farshad Arvin, Ali Emre Turgut, Tomáš Krajník, Salar Rahimi, Ilkin Ege Okay, Shigang Yue, Simon Watson, and Barry Lennox. ϕ clust: Pheromone-based aggregation for robotic swarms. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4288–4294. IEEE, 2018.
- [24] Farshad Arvin, Simon Watson, Ali Emre Turgut, Jose Espinosa, Tomáš Krajník, and Barry Lennox. Perpetual Robot Swarm: Long-term Autonomy of Mobile Robots Using on-the-fly Inductive Charging. *Journal of Intelligent & Robotic Systems*, pages 1–18, 2017.
- [25] Alessandra Bagnato, Regina Krisztina Bíró, Dario Bonino, Claudio Pastrone, Wilfried Elmenreich, René Reiners, Melanie Schranz, and Edin Arnautovic. Designing swarms of cyber-physical systems: The H2020 CPSwarm project. In *Proceedings of the Computing Frontiers Conference*, pages 305–312, 2017.
- [26] Mazen Bahaidarah, Fatemeh Rekabi Bana, Ali Emre Turgut, Ognjen Marjanovic, and Farshad Arvin. Optimization of a self-organized collective motion in a robotic swarm. In *International Conference on Swarm Intelligence*, pages 341–349. Springer, 2022.
- [27] Mazen Bahaidarah, Ognjen Marjanovic, Fatemeh Rekabi-Bana, and Farshad Arvin. An optimised robot swarm flocking with genetic algorithm. In *Proceedings of the IEEE International Conference on Mechatronics and Automation*, pages 1823–1828, 2023.

- [28] Mazen Bahaidarah, Fatemeh Rekabi-Bana, Ognjen Marjanovic, and Farshad Arvin. Swarm flocking using optimisation for a self-organised collective motion. *Swarm and Evolutionary Computation*, page 101491, 2024.
- [29] Zhe Ban, Junyan Hu, Barry Lennox, and Farshad Arvin. Self-organised collision-free flocking mechanism in heterogeneous robot swarms. *Mobile Networks and Applications*, pages 1–11, 2021.
- [30] Cyrill Baumann and Alcherio Martinoli. A modular functional framework for the design and evaluation of multi-robot navigation. *Robotics and Autonomous Systems*, 144:103849, 2021.
- [31] Levent Bayindir. A review of swarm robotics tasks. *Neurocomputing*, 172:292–321, 2016.
- [32] Levent Bayindir and Erol Şahin. A review of studies in swarm robotics. *Turkish Journal of Electrical Engineering & Computer Sciences*, 15(2):115–147, 2007.
- [33] Gerardo Beni and Jing Wang. Swarm intelligence in cellular robotic systems. In *Proceedings of the NATO Advanced Workshop on Robots and Biological Systems*, pages 703–712, 1989.
- [34] Amel Bennaceur, Carlo Ghezzi, Kenji Tei, Timo Kehler, Danny Weyns, Radu Calinescu, Schahram Dustdar, Zhenjiang Hu, Shinichi Honiden, Fuyuki Ishikawa, Zhi Jin, Jeffrey Kramer, Marin Litoiu, Michele Loreti, Gabriel Moreno, Hausi Muller, Laura Nenzi, Bashar Nuseibeh, Liliana Pasquale, Wolfgang Reisig, Heinz Schmidt, Christos Tsigkanos, and Haiyan Zhao. Modelling and analysing resilient cyber-physical systems. In *Proceedings of the International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 70–76. IEEE, 2019.
- [35] S Binitha, S Siva Sathya, et al. A Survey of Bio Inspired Optimization Algorithms. *International Journal of Soft Computing and Engineering*, 2(2):137–151, 2012.
- [36] Christian Blum and Xiaodong Li. Swarm Intelligence in Optimization. In Christian Blum and Daniel Merkle, editors, *Swarm Intelligence: Introduction and Applications*. Springer, 2008.
- [37] Michael Bodi, Ronald Thenius, Martina Szopek, Thomas Schmickl, and Karl Crailsheim. Interaction of robot swarms using the honeybee-inspired control algorithm beeclust. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):87–100, 2012.
- [38] Eric Bonabeau. Editor’s introduction: stigmergy. *Artificial Life*, 5(2):95–96, 1999.

- [39] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford university press, 1999.
- [40] Eric Bonabeau and Christopher Meyer. Swarm intelligence: A whole new way to think about business. *Harvard Business Review*, 79(5):106–115, 2001.
- [41] Tilman Börgers. *An Introduction to the Theory of Mechanism Design*. Oxford University Press, 2015.
- [42] Terry Bossomaier, Lionel Barnett, and Michael Harré. Information and phase transitions in socio-economic systems. *Complex Adaptive Systems Modeling*, 1(1):9, 2013.
- [43] Laszlo Böszörményi, Manfred del Fabro, Marian Kogler, Mathias Lux, O. Marques, and Anita Sobe. Innovative Directions in Self-organized Distributed Multimedia Systems. *Multimedia Tools and Applications, Springer*, 51(2):525–553, 2011.
- [44] Ali Önder Bozdoğan, Asım Egemen Yilmaz, and Murat Efa. Performance analysis of swarm optimization approaches for the generalized assignment problem in multi-target tracking applications. *Turkish Journal of Electrical Engineering and Computer Sciences*, 18(6):1059–1078, 2010.
- [45] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.
- [46] David Brandt, David Johan Christensen, and Henrik Hautop Lund. ATRON robots: Versatility from Self-reconfigurable Modules. In *Proceedings of the International Conference on Mechatronics and Automation*, pages 26–32, 2007.
- [47] Kyle Brown, Oriana Peltzer, Martin A Sehr, Mac Schwager, and Mykel J Kochenderfer. Optimal sequential task assignment and path finding for multi-agent robotic assembly planning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 441–447. IEEE, 2020.
- [48] Arne Brutschy, Giovanni Pini, Carlo Pinciroli, Mauro Birattari, and Marco Dorigo. Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous agents and multi-agent systems*, 28(1):101–125, 2014.
- [49] Jonathan Cacace, Riccardo Caccavale, Alberto Finzi, and Vincenzo Lippiello. Attentional multimodal interface for multi-drone search in the alps. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 1178–1183, 2016.

- [50] Guowei Cai, Jorge Dias, and Lakmal Seneviratne. A Survey of Small-Scale Unmanned Aerial Vehicles: Recent Advances and Future Development Trends. *Unmanned Systems*, 2(2):175–199, 2014.
- [51] Sonja Caldwell. Nodes – network & operation demonstration satellite. <https://www.nasa.gov/centers/ames/engineering/projects/nodes.html>, 2015. [Online; accessed 9-July-2019].
- [52] Scott Camazine. *Self-organization in biological systems*. Princeton University Press, 2003.
- [53] Scott Camazine, Nigel R. Franks, James Sneyd, Eric Bonabeau, Jean-Louis Deneubourg, and Guy Theraula. *Self-Organization in Biological Systems*. Princeton University Press, 2001.
- [54] Alexandre Campo, Álvaro Gutiérrez, Shervin Nouyan, Carlo Pinciroli, Valentin Longchamp, Simon Garnier, and Marco Dorigo. Artificial pheromone for path selection by a foraging swarm of robots. *Biological Cybernetics*, 103(5):339–352, 2010.
- [55] Gilles Caprari, Patrick Balmer, Ralph Piguët, and Roland Siegwart. The Autonomous Micro Robot “Alice”: A Platform for Scientific and Commercial Applications. In *Proceedings of the International Symposium on Micromechatronics and Human Science.-Creation of New Industry-(Cat. No. 98TH8388)*, pages 231–235. IEEE, 1998.
- [56] Stefano Carpin, Mike Lewis, Jijun Wang, Stephen Balakirsky, and Chris Scrapper. Usarsim: a robot simulator for research and education. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1400–1405. IEEE, 2007.
- [57] Andres Castano, Alberto Behar, and Peter M Will. The Conro Modules for Reconfigurable Robots. *IEEE/ASME transactions on mechatronics*, 7(4):403–409, 2002.
- [58] Eduardo Castello, Tomoyuki Yamamoto, Fabio Dalla Libera, Wenguo Liu, Alan F. T. Winfield, Yutaka Nakamura, and Hiroshi Ishiguro. Adaptive foraging for simulated and real robotic swarms: The dynamical response threshold approach. *Swarm Intelligence*, 10(1):1–31, 2016.
- [59] Shanzhi Chen, Jinling Hu, Yan Shi, Ying Peng, Jiayi Fang, Rui Zhao, and Li Zhao. Vehicle-to-everything (v2x) services supported by lte-based systems and 5g. *IEEE Communications Standards Magazine*, 1(2):70–76, 2017.
- [60] Ahmad Reza Cheraghi, Karol Actun, Sahdia Shahzad, and Kalman Graffi. Swarm-sim: A 2d & 3d simulation core for swarm agents. In *2020 3rd International Conference on Intelligent Robotic and Control Engineering (IRCE)*, pages 1–10. IEEE, 2020.

- [61] Chafika Chettaoui, Franck Delaplace, Pierre Lescanne, Mun'delanj Vestergaard, and René Vestergaard. Rewriting game theory as a foundation for state-based models of gene regulation. In *International Conference on Computational Methods in Systems Biology*, pages 257–270. Springer, 2006.
- [62] Anders Lyhne Christensen, Sancho Oliveira, Octavian Postolache, Maria João de Oliveira, Susana Sargento, Pedro Santana, Luís Nunes, Fernando J. Velez, Pedro Sebastião, Vasco Costa, Miguel Duarte, Jorge C. Gomes, Tiago Rodrigues, and Fernando Silva. Design of Communication and Control for Swarms of Aquatic Surface Drones. In *Proceedings of the International Conference on Agents and Artificial Intelligence*, pages 548–555, 2015.
- [63] Anders Lyhne Christensen, Rehan O’Grady, and Marco Dorigo. From fireflies to fault-tolerant swarms of robots. *Transactions on Evolutionary Computation*, 13(4):754–766, August 2009.
- [64] Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors. *Handbook of model checking*. Springer, 2016.
- [65] Christopher W. Cleghorn. *Particle Swarm Optimization: Empirical and Theoretical Stability Analysis*. PhD thesis, Department of Computer Science, University of Pretoria, South-Africa, 2017.
- [66] Christopher W. Cleghorn and Andries P. Engelbrecht. Particle swarm variants: Standardized convergence analysis. *Swarm Intelligence*, 9(2–3):177–203, 2015.
- [67] Isaac Cohen, David Corman, Jim Davis, Himanshu Khurana, Pieter J. Mosterman, and Stormo Lonny Prasad Venkatesh. Strategic opportunities for 21st century cyber-physical systems. Technical report, NSF, Steering Committee for Foundations in Innovation for Cyber-Physical Systems, 2012.
- [68] Steve Cook, Conrad Bock, Pete Rivett, Tom Rutt, Ed Seidewitz, Bran Selic, and Doug Tolbert. Unified modeling language (UML) version 2.5.1. Standard, Object Management Group (OMG), December 2017.
- [69] Nikolaus Correll and Alcherio Martinoli. Robust distributed coverage using a swarm of miniature robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 379–384, 2007.
- [70] Jorge Cortés and Magnus Egerstedt. Coordinated control of multi-robot systems: A survey. *SICE Journal of Control, Measurement, and System Integration*, 10(6):495–503, 2017.

- [71] Micael S. Couceiro, Carlos M. Figueiredo, David Portugal, Rui P. Rocha, and Nuno M. F. Ferreira. Initial deployment of a robotic team - a hierarchical approach under communication constraints verified on low-cost platforms. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4614–4619, 2012.
- [72] Bruno Courcelle. Graph rewriting: An algebraic and logic approach. In *Formal Models and Semantics*, pages 193–242. Elsevier, 1990.
- [73] Iain D Couzin, Jens Krause, Richard James, Graeme D Ruxton, and Nigel R Franks. Collective memory and spatial sorting in animal groups. *Journal of theoretical biology*, 218(1):1–11, 2002.
- [74] Felipe Cucker and Cristián Huepe. Flocking with informed agents. *Mathematics in Action*, 1(1):1–25, 2008.
- [75] Renan da Silva Tchilian, Ubirajara Franco Moreno, and Mariana Netto. Assisted teleoperation for a human-swarm interaction system. In *3rd IFAC Workshop on Cyber-Physical & Human Systems CPHS*, pages 602–607, 2020.
- [76] Christopher Dabrowski. Catastrophic event phenomena in communication networks: A survey. *Computer Science Review*, 18:10–45, 2015.
- [77] Simone D’Amico, Marco Pavone, Shailendhar Saraf, Abdulaziz Alhussien, Turki Al-Saud, Sasha Buchman, Robert Byer, and Charbel Farhat. Miniaturized autonomous distributed space system for future science and exploration. In *Proceedings of the International Workshop on Satellite Constellations and Formation Flying*, pages 1–20, 2015.
- [78] Kai Daniel, Bjoern Dusza, Andreas Lewandowski, and Christian Wietfeld. Airshield: A system-of-systems muav remote sensing architecture for disaster response. In *Proceedings of the 3rd Annual IEEE Systems Conference*, pages 196–200. IEEE, 2009.
- [79] Christian Darabos, Mario Giacobini, and Marco Tomassini. Performance and robustness of cellular automata computation on irregular networks. *Advances in Complex Systems*, 10(supp01):85–110, 2007.
- [80] Geert De Cubber, Daniela Doroftei, Daniel Serrano, Keshav Chintamani, Rui Sabino, and Stephane Ourevitch. The EU-ICARUS project: Developing assistive robotic tools for search and rescue operations. In *Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 1–4, 2013.
- [81] Kevin DeMarco, Eric Squires, Michael Day, and Charles Pippin. Simulating collaborative robots in a massive multi-agent game environment (scrimmage). In *Distributed Autonomous Robotic Systems*, pages 283–297. Springer, 2019.

- [82] Neel Dhanaraj, Nathan Hewitt, Casey Edmonds-Estes, Rachel Jarman, J. Seo, Henry Gunner, Alexandra Hatfield, Tucker Johnson, Lunet Yifru, Julietta Maffeo, Guilherme Pereira, Jason Gross, and Yu Gu. Adaptable platform for interactive swarm robotics (APIS): A human-swarm interaction research testbed. In *Proceedings of the 19th International Conference on Advanced Robotics*, pages 720–726, 2019.
- [83] Elisa Donati, Godfried J van Vuuren, Katsuaki Tanaka, Donato Romano, Thomas Schmickl, and Cesare Stefanini. amussels: Diving and anchoring in a new bio-inspired under-actuated robot class for long-term environmental exploration and monitoring. In *Proceedings of the Conference Towards Autonomous Robotic Systems*, pages 300–314. Springer, 2017.
- [84] Marco Dorigo and Mauro Birattari. Ant colony optimization. In *Encyclopedia of machine learning*, pages 36–39. Springer, 2011.
- [85] Marco Dorigo, Mauro Birattari, and Manuele Brambilla. Swarm robotics. *Scholarpedia*, 9(1):1463, 2014.
- [86] Marco Dorigo, Mauro Birattari, et al. Swarm intelligence. *Scholarpedia*, 2(9):1462, 2007.
- [87] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.
- [88] Marco Dorigo, Mauro Birattari, and Thomas Stützle. Ant colony optimization: A computational intelligence technique. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006.
- [89] Marco Dorigo, Dario Floreano, Luca Maria Gambardella, Francesco Mondada, Stefano Nolfi, Tarek Baaboura, Mauro Birattari, Michael Bonani, Manuele Brambilla, Arne Brutschy, et al. Swarmanoid: A Novel Concept for the Study of Heterogeneous Robotic Swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71, 2013.
- [90] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. A Bradford Book, The MIT Press, 2004.
- [91] Marco Dorigo, Guy Theraulaz, and Vito Trianni. Reflections on the future of swarm robotics. *Science robotics*, 5(49), December 2020.
- [92] Barzin Doroodgar, Maurizio Ficocelli, Babak Mobedi, and Goldie Nejat. The search for survivors: Cooperative human-robot interaction in search and rescue environments using semi-autonomous robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2858–2863, 2010.

- [93] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.
- [94] Miguel Duarte, Vasco Costa, Jorge Gomes, Tiago Rodrigues, Fernando Silva, Sancho Moura Oliveira, and Anders Lyhne Christensen. Evolution of Collective Behaviors for a Real Swarm of Aquatic Surface Robots. *PloS one*, 11(3):25, 2016.
- [95] Frederick Ducatelle, Gianni A Di Caro, Alexander Förster, Michael Bonani, Marco Dorigo, Stéphane Magnenat, Francesco Mondada, Rehan O’Grady, Carlo Pinciroli, Philippe Régnier, et al. Cooperative navigation in robotic swarms. *Swarm Intelligence*, 8(1):1–33, 2014.
- [96] David Duff, Mark Yim, and Kimon Roufas. Evolution of Polybot: A Modular Reconfigurable Robot. In *Proceedings of the Harmonic Drive Intelligent Symposium*, 2001.
- [97] Wilfried Elmenreich and Hermann de Meer. Self-organizing networked systems for technical applications: A discussion on open issues. In *Proceedings of the 3rd International Workshop on Self-Organizing Systems*, pages 1–9. Springer, 2008.
- [98] Wilfried Elmenreich, Raissa D’Souza, Christian Bettstetter, and Hermann de Meer. A survey of models and design methods for self-organizing networked systems. In *Proceedings of the 4th International Workshop on Self-Organizing Systems*, volume LNCS 5918, page 37–49. Springer, 2009.
- [99] Wilfried Elmenreich, Tobias Ibounig, and Istvan Fehervari. Robustness versus Performance in Sorting and Tournament Algorithms. *Acta Polytechnica*, 6(5):7–18, 2009.
- [100] Wilfried Elmenreich and Gernot Klingler. Genetic Evolution of a Neural Network for the Autonomous Control of a Four-Wheeled Robot. In *Proceedings of the 6th Mexican International Conference on Artificial Intelligence - Special Session (MICAI)*, pages 396–406, 2007.
- [101] Wilfried Elmenreich, Alexander Schnabl, and Melanie Schranz. An artificial hormone-based algorithm for production scheduling from the bottom-up. In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence*. SciTePress, 2021.
- [102] Wilfried Elmenreich and Stefan Schuster. Demand response by decentralized device control based on voltage level. In *Proceedings of the 7th International Workshop on Self-Organizing Systems*, pages 186–189. Springer Verlag, 2013.
- [103] Thomas Eltz. Tracing pollinator footprints on natural flowers. *Journal of Chemical Ecology*, 32:907–915, 2006.

- [104] Sebastian Engell, Radoslav Paulen, Michel A Reniers, Christian Sonntag, and Haydn Thompson. Core research and innovation areas in cyber-physical systems of systems. In *Proceedings of the International Workshop on Design, Modeling, and Evaluation of Cyber Physical Systems*, pages 40–55. Springer, 2015.
- [105] Adam Erskine, Thomas Joyce, and J. Michael Herrmann. Stochastic stability of particle swarm optimisation. *Swarm Intelligence*, 11(3):295–315, 2017.
- [106] Kasra Eshaghi, Yuchen Li, Zendai Kashino, Goldie Nejat, and Beno Benhabib. mroberto 2.0 – an autonomous millirobot with enhanced locomotion for swarm robotics. *IEEE Robotics and Automation Letters*, 5(2):962–969, 2020.
- [107] Hadi Eskandar, Ali Sadollah, Ardeshir Bahreininejad, and Mohd Hamdi. Water cycle algorithm – a novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers & Structures*, 110:151–166, 2012.
- [108] Tingxiang Fan, Pinxin Long, Wenxi Liu, and Jia Pan. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *The International Journal of Robotics Research*, 39(7):856–892, 2020.
- [109] Jolyon J Faria, John RG Dyer, Colin R Tosh, and Jens Krause. Leadership and social information use in human crowds. *Animal Behaviour*, 79(4):895 – 901, 2010.
- [110] B. G. Farley and W. Clark. Simulation of self-organizing systems by digital computer. *Transactions of the IRE Professional Group on Information Theory*, 4(4):76–84, 1954.
- [111] István Fehérvári and Wilfried Elmenreich. Evolving neural network controllers for a team of self-organizing robots. *Journal of Robotics*, 2010.
- [112] István Fehérvári and Wilfried Elmenreich. Evolution as a tool to design self-organizing systems. In *Proceedings of the IFIP 7th International Workshop on Self-Organizing Systems*, pages 139–144, 2013.
- [113] Ouajdi Felfoul, Mahmood Mohammadi, Samira Taherkhani, Dominic De Lanauze, Yong Zhong Xu, Dumitru Loghin, Sherief Essa, Sylwia Jancik, Daniel Houle, Michel Laffleur, et al. Magneto-aerotactic bacteria deliver drug-containing nanoliposomes to tumour hypoxic regions. *Nature Nanotechnology*, 11(11), 7 p., 2016.
- [114] Jean-Claude Fernandez, Laurent Mounier, and Cyril Pachon. A model-based approach for robustness testing. In *IFIP International Conference on Testing of Communicating Systems*, pages 333–348. Springer, 2005.

- [115] Eliseo Ferrante, Edgar Duéñez Guzmán, Ali Emre Turgut, and Tom Wenseleers. Geswarm: Grammatical evolution for the automatic synthesis of collective behaviors in swarm robotics. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pages 17–24, 2013.
- [116] Eliseo Ferrante, Ali Emre Turgut, Marco Dorigo, and Cristian Huepe. Collective motion dynamics of active solids and active crystals. *New Journal of Physics*, 15(9):095011, 2013.
- [117] Eliseo Ferrante, Ali Emre Turgut, Marco Dorigo, and Cristián Huepe. Elasticity-driven collective motion in active solids and active crystals. *arXiv preprint arXiv:1301.2620*, 2013.
- [118] Eliseo Ferrante, Ali Emre Turgut, Cristián Huepe, Alessandro Stranieri, Carlo Pinciroli, and Marco Dorigo. Self-organized flocking with a mobile robot swarm: a novel motion control method. *Adaptive Behavior*, 20(6):460–477, 2012.
- [119] Eliseo Ferrante, Ali Emre Turgut, Alessandro Stranieri, Carlo Pinciroli, Mauro Birattari, and Marco Dorigo. A self-adaptive communication strategy for flocking in stationary and non-stationary environments: complete data. *Supplementary information [http://iridia. ulb. ac. be/supp/IridiaSupp2011-025/](http://iridia.ulb.ac.be/supp/IridiaSupp2011-025/)*. Accessed, 6, 2013.
- [120] Stanley Fields. Phermone response in yeast. *Trends in Biochemical Sciences*, 15(7):270–273, 1990.
- [121] Apache Flink. <https://flink.apache.org/>. [Online; accessed 8-August-2024].
- [122] Dario Floreano and Claudio Mattiussi. *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. MIT Press, 2008.
- [123] National Science Foundation. Cyber Physical Systems. https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503286. [Online; accessed 15-March-2018].
- [124] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Vito Trianni, and Mauro Birattari. Automode: A novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*, 8(2):89–112, 2014.
- [125] Ryusuke Fujisawa, Shigeto Dobata, Ken Sugawara, and Fumitoshi Matsuno. Designing pheromone communication in swarm robotics: Group foraging behavior mediated by chemical substance. *Swarm Intelligence*, 8(3):227–246, 2014.

- [126] David Furlonger. Swarm intelligence: From smart cars to smart traffic, BusinessLIVE, August 2017. <https://www.businesslive.co.za/bd/life/motoring/2017-08-14-swarm-intelligence-from-smart-cars-to-smart-traffic>, 2017. [Online; accessed 27-September-2019].
- [127] Epic Games. Unity Technologies. Unity. Available at: <https://unity.com/de> [Accessed: 26 March 2023].
- [128] Epic Games. Unreal Engine. <https://www.unrealengine.com/>, 2024. [Online; accessed 23-March-2023].
- [129] X. Z. Gao, V. Govindasamy, H. Xu, K. Wang, and K. Zenger. Harmony search method: Theory and applications. *Computational Intelligence and Neuroscience*, 258491, April 2015.
- [130] Michael R Garey, David S Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129, 1976.
- [131] Simon Garnier, Jacques Gautrais, and Guy Theraulaz. The biological principles of swarm intelligence. *Swarm Intelligence*, 1(1):3–31, 2007.
- [132] Simon Garnier, Christian Jost, Jacques Gautrais, Masoud Asadpour, Gilles Caprari, Raphaël Jeanson, Anne Grimal, and Guy Theraulaz. The Embodiment of Cockroach Aggregation Behavior in a Group of Micro-robots. *Artificial life*, 14(4):387–408, 2008.
- [133] Simon Garnier, Faben Tache, Maud Combe, Anne Grimal, and Guy Theraulaz. Alice in pheromone land: An experimental setup for the study of ant-like robots. In *2007 IEEE Swarm Intelligence Symposium*, pages 37–44. IEEE, 2007.
- [134] Andrea Gasparri, Giuseppe Oriolo, Attilio Priolo, and Giovanni Ulivi. A swarm aggregation algorithm for multi-robot systems based on local interaction. In *Proceedings of the IEEE International Conference on Control Applications*, pages 1497–1502, 2012.
- [135] Alexander J. Gates and Luis M. Rocha. Control of complex networks requires both structure and dynamics. *Nature, Scientific Reports*, 6(24456), 2016.
- [136] Veysel Gazi and Kevin M. Passino. Stability analysis of social foraging swarms. *IEEE Transactions on Systems, Man, and Cybernetics: Part B*, 34(1):539–557, 2004.
- [137] GCtronic Sagl. Elisa-3 robots - 38 units with obstacle avoidance. <https://www.youtube.com/watch?v=WDxfIFhpm1g>, 2014. [Online; accessed 30-January-2020].

- [138] Hwaiu Geng, editor. *Semiconductor Manufacturing Handbook*. McGraw-Hill Education, 2018.
- [139] Edd Gent. Ever dream of controlling robot swarms? This new virtual reality headset could help. *Science*, 2019. <https://www.science.org/content/article/ever-dream-controlling-robot-swarms-new-virtual-reality-headset-could-help>, [Online; accessed 21-December-2023].
- [140] Carlos Gershenson. *Design and control of self-organizing systems*. CopIt Arxivs, 2007.
- [141] Mohammadreza Ghorbaniparvar. Survey on forced oscillations in power system. *Journal of Modern Power Systems and Clean Energy*, 5:671–682, June 2017.
- [142] S. N. Givigi and H. M. Schwartz. A game theoretic approach to swarm robotics. *Applied Bionics and Biomechanics*, 3(3):131–142, 2006.
- [143] Alan Godoy, Pedro Tabacof, and Fernando J. Von Zuben. The role of the interaction network in the emergence of diversity of behavior. *PLoS ONE*, 12(2):e0172073, 2017.
- [144] Jorge Gomes, Paulo Urbano, and Anders Lyhne Christensen. Evolution of swarm robotics systems with novelty search. *Swarm Intelligence*, 7(2):115–144, 2013.
- [145] David Green, Aldeida Aleti, and Julian Garcia. The nature of nature: Why nature-inspired algorithms work. In Srikanta Patnaik, Xin-She Yang, and Kazumi Nakamatsu, editors, *Nature-Inspired Computing and Optimization: Theory and Applications*, pages 1–27. Springer, 2017.
- [146] Calvin Gregory and Andrew Vardy. microUSV: A low-cost platform for indoor marine swarm robotics research. *HardwareX*, 7:e00105, 2020.
- [147] Kay S. Grennan, Chao Chen, Elliot S. Gershon, and Chunyu Liu. Molecular network analysis enhances understanding of the biology of mental disorders. *Bioessays*, 36(6):606–616, 2014.
- [148] Boris Gromov, Luca Gambardella, and Gianni A. Di Caro. Wearable multi-modal interface for human multi-robot interaction. In *Proceedings of the 14th IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 240–245, 2016.
- [149] Roderich Groß, Michael Bonani, Francesco Mondada, and Marco Dorigo. Autonomous Self-Assembly in Swarm-Bots. *IEEE Transactions on Robotics*, 22(6):1115–1130, 2006.
- [150] D. Grossman, I. S. Aranson, and E. Ben Jacob. Emergence of agent swarm migration and vortex formation through inelastic collisions. *New Journal of Physics*, 10(2):023036, 2008.

- [151] Chathika Gunaratne and Ivan Garibay. NL4Py: Agent-based modeling in Python with parallelizable NetLogo workspaces. *SoftwareX*, 16:100801, 2021.
- [152] Kexin Guo, Zhirong Qiu, Cunxiao Miao, Abdul Hanif Zaini, Chun-Lin Chen, Wei Meng, and Lihua Xie. Ultra-wideband-based localization for quadcopter navigation. *Unmanned Systems*, 4(01):23–34, 2016.
- [153] Heiko Hamann. *Swarm robotics: A formal approach*. Springer, 2018.
- [154] Kyle Robert Harrison, Andries P. Engelbrecht, and Beatrice M. Ombuki-Berman. Self-adaptive particle swarm optimization: A review and analysis of convergence. *Swarm Intelligence*, 12(3):187–226, 2018.
- [155] Aboul Ella Hassanien and Eid Alamry. *Swarm Intelligence: Principles, Advances, and Applications*. CRC Press, 2015.
- [156] Sabine Hauert, Jean-Christophe Zufferey, and Dario Floreano. Evolved swarming without positioning information: An application in aerial communication relay. *Autonomous Robots*, 26(1):21–32, 2009.
- [157] Apache HBase. <https://hbase.apache.org/>. [Online; accessed 8-August-2024].
- [158] Anni Heckert. *Entwicklung eines dynamischen Modells und Parameterschätzung für den FINken 3 Quadcopter*. Bachelor’s thesis, Fakultät für Elektrotechnik und Informationstechnik, Otto-von-Guericke-Universität Magdeburg, 2016.
- [159] Herianto and D. Kurabayashi. Realization of an artificial pheromone system in random data carriers using rfid tags for autonomous navigation. In *2009 IEEE International Conference on Robotics and Automation*, pages 2288–2293, May 2009.
- [160] Francis Heylighen. The science of self-organization and adaptivity. *The Encyclopedia of Life Support Systems*, 5(3):253–280, 2001.
- [161] Francis Heylighen. Complexity and self-organization, 2008.
- [162] Nicholas R. Hoff, Amelia Sagoff, Robert J. Wood, and Radhika Nagpal. Two foraging algorithms for robot swarms using only local communication. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, pages 123–130, 2010.
- [163] John Henry Holland et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.
- [164] Richard Holzer and Hermann De Meer. On modeling of self-organizing systems. In *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems*, pages 1–6, 2008.

- [165] Richard Holzer and Hermann De Meer. Quantitative modeling of self-organizing properties. In *International Workshop on Self-Organizing Systems*, pages 149–161. Springer, 2009.
- [166] Matthew B Hoy. Alexa, siri, cortana, and more: An introduction to voice assistants. *Medical Reference Services Quarterly*, 37(1):81–88, 2018.
- [167] Allen Hsu, Huihua Zhao, Martin Gaudreault, Annjoe Wong Foy, and Ron Pelrine. Magnetic milli-robot swarm platform: A safety barrier certificate enabled, low-cost test bed. *IEEE Robotics and Automation Letters*, 5(2):2913–2920, 2020.
- [168] Junyan Hu, Parijat Bhowmick, Farshad Arvin, Alexander Lanzon, and Barry Lennox. Cooperative control of heterogeneous connected vehicle platoons: An adaptive leader-following approach. *IEEE Robotics and Automation Letters*, 5(2):977–984, 2020.
- [169] Junyan Hu, Parijat Bhowmick, Inmo Jang, Farshad Arvin, and Alexander Lanzon. A decentralized cluster formation containment framework for multirobot systems. *IEEE Transactions on Robotics*, 2021.
- [170] Junyan Hu, Hanlin Niu, Joaquin Carrasco, Barry Lennox, and Farshad Arvin. Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(12):14413–14423, 2020.
- [171] Junyan Hu, Ali Emre Turgut, Tomáš Krajník, Barry Lennox, and Farshad Arvin. Occlusion-based coordination protocol design for autonomous robotic shepherding tasks. *IEEE Transactions on Cognitive and Developmental Systems*, 2020.
- [172] Junyan Hu, Ali Emre Turgut, Barry Lennox, and Farshad Arvin. Robust formation coordination of robot swarms with nonlinear dynamics and unknown disturbances: Design and experiments. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2021.
- [173] Ye Huang, Amos Brocco, Pierre Kuonen, Michèle Courant, and Béat Hirsbrunner. Smartgrid: A fully decentralized grid scheduling framework supported by swarm intelligence. In *Proceedings of the 7th International Conference on Grid and Cooperative Computing*, pages 160–168, 2008.
- [174] Mimmo Iannelli and Andrea Pugliese. *An Introduction to Mathematical Population Dynamics: A long the trail of Volterra and Lotka*. Springer, 2014.
- [175] Infineon Technologies AG. <https://www.infineon.com/>. [Online; accessed 22-March-2024].

- [176] Malek Itani, Tuochao Chen, Takuya Yoshioka, and Shyamnath Gollakota. Creating speech zones with self-distributing acoustic swarms. *Nature Communications*, 14(1):5684, 2023.
- [177] Duncan E. Jackson and Francis L.W. Ratnieks. Communication in ants. *Current Biology*, 16(15):R570 – R574, 2006.
- [178] Inmo Jang, Junyan Hu, Farshad Arvin, Joaquin Carrasco, and Barry Lennox. Omnipotent virtual giant for remote human–swarm interaction. In *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, pages 488–494. IEEE, 2021.
- [179] Fredrik Jansson, Matthew Hartley, Martin Hinsch, Ivica Slavkov, Noemí Carranza, Tjelvar SG Olsson, Roland M Dries, Johanna H Grönqvist, Athanasius FM Marée, James Sharpe, et al. Kilombo: a kilobot simulator to enable effective research in swarm robotics. *arXiv preprint arXiv:1511.04285*, 2015.
- [180] Midhat Jdeed, Arthur Pitman, and Wilfried Elmenreich. Spiderino - a low cost robot for swarm research and educational purposes, technical documentation. Technical report, Institute of Networked and Embedded Systems, Alpen-Adria-Universität Klagenfurt, Austria, 2018.
- [181] Midhat Jdeed, Sergii Zhevzhyk, Florian Steinkellner, and Wilfried Elmenreich. Spiderino: A low-cost Robot for Swarm Research and Educational Purposes. In *Proceedings of the 13th Workshop on Intelligent Solutions in Embedded Systems*, pages 35–39, 2017.
- [182] Shu Jiang and Ronald C Arkin. Mixed-initiative human-robot interaction: Definition, taxonomy, and survey. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 954–961, 2015.
- [183] Yaochu Jin and Jurgen Branke. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.
- [184] Martin Jinek, Krzysztof Chylinski, Ines Fonfara, Michael Hauer, Jennifer A. Doudna, and Emmanuelle Charpentier. A programmable dual-RNA-guided DNA endonuclease in adaptive bacterial immunity. *Science*, 337(6096):816–821, 2012.
- [185] Geraint Jones, Nadia Berthouze, Roman Bielski, and Simon Julier. Towards a situated, multimodal interface for multiple UAV control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1739–1744, 2010.
- [186] Simon Jones, Emma Milner, Mahesh Sooriyabandara, and Sabine Hauert. Dots: An open testbed for industrial swarm robotic solutions. *arXiv preprint arXiv:2203.13809*, 2022.

- [187] Simon Jones, Matthew Studley, Sabine Hauert, and Alan Frank Thomas Winfield. A two Teraflop Swarm. *Frontiers in Robotics and AI*, 5:11, 2018.
- [188] Visakan Kadirkamanathan, Kirusnapillai Selvarajah, and Peter J. Fleming. Stability analysis of the particle dynamics in particle swarm optimizer. *IEEE Transactions on Evolutionary Computation*, 10(3), 2006.
- [189] Apache Kafka. <https://kafka.apache.org>. [Online; accessed 8-August-2024].
- [190] Fumio Kanehiro, Hirohisa Hirukawa, and Shuuji Kajita. Openhrp: Open architecture humanoid robotics platform. *The International Journal of Robotics Research*, 23(2):155–165, 2004.
- [191] Minoru Kanehisa, Susumu Goto, Miho Furumichi, Mao Tanabe, and Mika Hirakawa. KEGG for representation and analysis of molecular networks involving diseases and drugs. *Nucleic Acids Research*, 38(suppl.1):D355–D360, 2010.
- [192] Chang-Kwon Kang. Marsbee - Swarm of Flapping Wing Flyers for Enhanced Mars Exploration. https://www.nasa.gov/directorates/spacetech/niac/2018_Phase_I_Phase_II/Marsbee_Swarm_of_Flapping_Wing_Flyers_for_Enhanced_Mars_Exploration, 2018. [Online; accessed 05-September-2019].
- [193] Akbar Karimi, Hadi Nobahari, and Patrick Siarry. Continuous ant colony system and tabu search algorithms hybridized for global minimization of continuous multi-minima functions. *Computational Optimization and Applications*, 45(3):639–661, 2010.
- [194] P. Karlson and M Lüscher. ‘Pheromones’: A New Term for a Class of Biologically Active Substances. *Nature*, 183:55–56, 1959.
- [195] Michael N. Katehakis and Arthur F. Veinott. The multi-armed bandit problem: Decomposition and computation. *Mathematics of Operations Research*, 12(2):262–268, 1987.
- [196] Ali Kaveh and Neda Farhoudi. A new optimization method: Dolphin echolocation. *Advances in Engineering Software*, 59:53–70, 2013.
- [197] James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.
- [198] Serge Kernbach, Dagmar Häbe, Olga Kernbach, Ronald Thenius, Gerald Radspieler, Toshifumi Kimura, and Thomas Schmickl. Adaptive Collective Decision-making in Limited Robot Swarms without Communication. *The International Journal of Robotics Research*, 32(1):35–55, 2013.

- [199] Serge Kernbach, Eugen Meister, Florian Schlachter, Kristof Jebens, Marc Szymanski, Jens Liedke, Davide Laneri, Lutz Winkler, Thomas Schmickl, Ronald Thenius, et al. Symbiotic Robot Organisms: REPLICATOR and SYMBRION Projects. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, pages 62–69, 2008.
- [200] Alexander Kettler. Cleaning up with a Swarm of Robots. https://www.youtube.com/watch?v=FqMP_AIkDj8, 2012. [Online; accessed 05-February-2020].
- [201] Alexander Kettler, Marc Szymanski, and Heinz Wörn. The Wanda Robot and its Development System for Swarm Algorithms. In *Advances in Autonomous Mini Robots*, pages 133–146. Springer, 2012.
- [202] Ali Abdul Khaliq and Alessandro Saffiotti. Stigmergy at work: Planning and navigation for a service robot on an rfid floor. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1085–1092, May 2015.
- [203] Elnaz Khatmi, Wilfried Elmenreich, Kristina Wogatai, Melanie Schranz, Martina Umlauft, Walter Laure, and Andreas Wuttei. Swarm intelligence layer to control autonomous agents (SWILT). In *Proceedings of the Research Project Showcase at Software Technologies: Applications and Foundations (STAF-RPS19)*, 2019.
- [204] Kicksat. Kicksat: A tiny open source spacecraft project. <https://kicksat.github.io/>. [Online; accessed 9-July-2019].
- [205] Hannes Kirchhoff, Noara Kebir, Kirsten Neumann, Peter W. Heller, and Kai Strunz. Developing mutual success factors and their application to swarm electrification: microgrids with 100 % renewable energies in the Global South and Germany. *Journal of Cleaner Production*, 128:190–200, 2016.
- [206] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. In *Readings in Computer Vision*, pages 606–615. Elsevier, 1987.
- [207] Zalan Kiszli, Seongin Na, and Farshad Arvin. Toward a Myriad Robot Swarm Aggregation. In *7th International Conference on Control and Robotics Engineering*. IEEE, 2022.
- [208] John Klingner, Anshul Kanakia, Nicholas Farrow, Dustin Reishus, and Nikolaus Correll. A Stick-slip Omnidirectional Powertrain for low-cost Swarm Robotics: Mechanism, Calibration, and Control. *IEEE International Conference on Intelligent Robots and Systems*, pages 846–851, 2014.

- [209] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.
- [210] Andreas Kolling, Phillip Walker, Nilanian Chakraborty, Katia Sycara, and Michael Lewis. Human interaction with robot swarms: A survey. *IEEE Transactions on Human-Machine Systems*, 46(1):9–26, 2016.
- [211] Shigeru Kondo and Takashi Miura. Reaction-diffusion model as a framework for understanding biological pattern formation. *science*, 329(5999):1616–1620, 2010.
- [212] Tomáš Krajník, Matías Nitsche, Jan Faigl, Petr Vaněk, Martin Saska, Libor Preučil, Tom Duckett, and Marta Mejail. A practical multirobot localization system. *Journal of Intelligent & Robotic Systems*, 76(3-4):539–562, 2014.
- [213] Jens Krause, Graeme D Ruxton, and Stefan Krause. Swarm intelligence in animals and humans. *Trends in Ecology & Evolution*, 25(1):28 – 34, 2010.
- [214] Jonas Krause, Jelson Cordeiro, Rafael Stubs Parpinelli, and Heitor Silverio Lopes. A Survey of Swarm Algorithms Applied to Discrete Optimization. In Xin-She Yang, Zhihua Cui, Renbin Xiao, Amir Hossein Gandomi, and Mehmet Karamanoglu, editors, *Swarm Intelligence and Bio-Inspired Computation*, pages 169–191. Elsevier, 2013.
- [215] Stefan Krause, Richard James, Jolyon J Faria, Graeme D Ruxton, and Jens Krause. Swarm intelligence in humans: Diversity can trump ability. *Animal Behaviour*, 81(5):941 – 948, 2011.
- [216] Geert-Jan M Kruijff, M Janíček, Shanker Keshavdas, Benoit Larochele, Hendrik Zender, Nanja JJM Smets, Tina Mioch, Mark A Neerincx, Jurriaan Van Diggelen, Francis Colas, et al. Experience in system design for human-robot teaming in urban search and rescue. In *Field and Service Robotics*, pages 111–125, 2014.
- [217] Haruhisa Kurokawa, Akiya Kamimura, Eiichi Yoshida, Kohji Tomita, Shigeru Kokaji, and Satoshi Murata. M-TRAN II: Metamorphosis from a Four-legged Walker to a Caterpillar. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2454–2459, 2003.
- [218] Haruhisa Kurokawa, Kohji Tomita, Akiya Kamimura, Shigeru Kokaji, Takashi Hasuo, and Satoshi Murata. Distributed Self-reconfiguration of M-TRAN III Modular Robotic System. *The International Journal of Robotics Research*, 27(3-4):373–386, 2008.

- [219] Chang kwon Kang. Marsbee: Swarm of flapping wing flyers for enhanced Mars exploration. https://www.nasa.gov/directorates/spacetech/niac/2018_Phase_I_Phase_II/Marsbee_Swarm_of_Flapping_Wing_Flyers_for_Enhanced_Mars_Exploration, 2018. [Online; accessed 9-July-2019].
- [220] George Kyriakarakos and George Papadakis. Multispecies Swarm Electrification for Rural Areas of the Developing World. *Applied Sciences*, 9(19):3992, 2019.
- [221] Jean-Claude Laprie. From dependability to resilience. In *38th IEEE/IFIP Int. Conf. On dependable systems and networks*, pages G8–G9, 2008.
- [222] Eugene L Lawler, Jan Karel Lenstra, Alexander HG Rinnooy Kan, and David B Shmoys. Sequencing and scheduling: Algorithms and complexity. *Handbooks in Operations Research and Management Science*, 4:445–522, 1993.
- [223] Mathieu Le Goc, Lawrence H Kim, Ali Parsaei, Jean-Daniel Fekete, Pierre Dragicevic, and Sean Follmer. Zooids: Building Blocks for Swarm User Interfaces. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 97–109, 2016.
- [224] Edward A Lee. Cyber physical systems: Design challenges. In *Proceedings of the 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing*, pages 363–369, 2008.
- [225] Edward A. Lee. The past, present and future of cyber-physical systems: A focus on models. *Transactions on Cyber-Physical Systems*, 1(1):3:1–3:26, February 2017.
- [226] Edward Ashford Lee and Sanjit A Seshia. *Introduction to embedded systems: A cyber-physical systems approach*. MIT Press, 2016.
- [227] Lin Li, Ruochen Hao, Wanjing Ma, Xinzhou Qi, and Chenxue Diao. Swarm intelligence based algorithm for management of autonomous vehicles on arterials. Technical report, SAE, 2018.
- [228] Shimin Li, Huiling Chen, Mingjing Wang, Ali Asghar Heidari, and Seyedali Mirjalili. Slime mould algorithm: A new method for stochastic optimization. *Future Generation Computer Systems*, 111:300–323, 2020.
- [229] Sean Lim, Shiyi Wang, Barry Lennox, and Farshad Arvin. Beeground—an open-source simulation platform for large-scale swarm robotics applications. In *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*, pages 75–79. IEEE, 2021.

- [230] Marcelo A. Limeira, Luis Piardi, Vivian Cremer Kalempa, André Schneider de Oliveira, and Paulo Leitão. Wsbot: A tiny, low-cost swarm robot for experimentation on industry 4.0. In *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, pages 293–298, 2019.
- [231] Tian Liu, Xuelong Sun, Cheng Hu, Qinbing Fu, and Shigang Yue. A multiple pheromone communication system for swarm intelligence. *IEEE Access*, 9:148721–148737, 2021.
- [232] Yang-Yu Liu, Jean-Jacques Slotine, and Albert-László Barabási. Observability of complex systems. *Proceedings of the National Academy of Sciences*, 110(7):2460–2465, 2013.
- [233] Zheyu Liu, Craig West, Barry Lennox, and Farshad Arvin. Local bearing estimation for a swarm of low-cost miniature robots. *Sensors*, 20(11):3308, 2020.
- [234] Yuri K. Lopes, Stefan M. Trenkwalder, André B. Leal, Tony J. Dodd, and Roderich Groß. Supervisory control theory applied to swarm robotics. *Swarm Intelligence*, 10(1):65–97, 2016.
- [235] Manuel López-Ibáñez and Thomas Stützle. An experimental analysis of design choices of multi-objective ant colony optimization algorithms. *Swarm Intelligence*, 6(3):207–232, 2012.
- [236] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, and Keith Sullivan. MASON: A new multi-agent simulation toolkit. In *Proceedings of the 2004 SwarmFest Workshop*, pages 316–327, Ann Arbor, Michigan, USA, May 2004.
- [237] Arun Mahadev, Dominik Krupke, Sandor P. Fekete, and Aaron T. Becker. Mapping and coverage with a particle swarm controlled by uniform inputs. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1097–1104, 2017.
- [238] Lorenzo Marconi, Claudio Melchiorri, Michael Beetz, Dejan Pangercic, Roland Siegwart, Stefan Leutenegger, Raffaella Carloni, Stefano Stramigioli, Herman Bruyninckx, Patrick Doherty, et al. The SHERPA project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments. In *Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 1–4, 2012.
- [239] David Masad and Jacqueline Kazil. Mesa: an agent-based modeling framework. In *14th PYTHON in Science Conference*, volume 2015, pages 53–60. Citeseer, 2015.

- [240] Ralf Mayet, Jonathan Roberz, Thomas Schmickl, and Karl Crailsheim. Antbots: A feasible visual emulation of pheromone trails for swarm robots. In *Proceedings of the International Conference on Swarm Intelligence*, pages 84–94. Springer, 2010.
- [241] Cassandra McCord, Jorge Pena Queralta, Tuan Nguyen Gia, and Tomi Westerlund. Distributed progressive formation control for multi-agent systems: 2d and 3d deployment of uavs in ros/gazebo with rotors. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE, 2019.
- [242] James McLurkin, Andrew J Lynch, Scott Rixner, Thomas W Barr, Alvin Chou, Kathleen Foster, and Siegfried Bilstein. A low-cost Multi-robot System for Research, Teaching, and Outreach. In *Distributed Autonomous Robotic Systems*, pages 597–609. Springer, 2013.
- [243] Andrew McNabb, Matthew Gardner, and Kevin Seppi. An exploration of topologies and communication in large particle swarms. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 712–719, 2009.
- [244] Andreas M Menzel and Takao Ohta. Soft deformable self-propelled particles. *EPL (Europhysics Letters)*, 99(5):58001, 2012.
- [245] Vera Mersheeva. *UAV Routing Problem for Area Monitoring in a Disaster Situation*. Ph.d. dissertation, Universität Klagenfurt, 2015. Dissertation.
- [246] Olivier Michel. Cyberbotics ltd. webots™: professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):5, 2004.
- [247] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in engineering software*, 69:46–61, 2014.
- [248] Melanie Mitchell. *Complexity: A guided tour*. Oxford University Press, 2009.
- [249] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, pages 59–65, 2009.
- [250] Francesco Mondada, Michael Bonani, Fanny Riedo, Manon Briod, Léa Pereyre, Philippe Réturnaz, and Stéphane Magnenat. Bringing robotics to formal education: The thymio open-source hardware robot. *IEEE Robotics & Automation Magazine*, 24(1):77–85, 2017.

- [251] Francesco Mondada, Alcherio Martinoli, Nicolaus Correll, Alexey Gribovskiy, José Ignacio Halloy, Roland Siegwart, and Jean-Louis Deneubourg. A general methodology for the control of mixed natural-artificial societies. *Handbook of Collective Robotics*, pages 399–428, 2011.
- [252] Francesco Mondada, Giovanni C Pettinaro, Andre Guignard, Ivo W Kwee, Dario Floreano, Jean-Louis Deneubourg, Stefano Nolfi, Luca Maria Gambardella, and Marco Dorigo. Swarm-bot: A new distributed robotic concept. *Autonomous robots*, 17(2-3):193–221, 2004.
- [253] Francesco Mondada, Giovanni Cosimo Pettinaro, Ivo W Kwee, André Guignard, Luca Maria Gambardella, Dario Floreano, Stefano Nolfi, Jean-Louis Deneubourg, and Marco Dorigo. SWARM-BOT: A Swarm of Autonomous Mobile Robots with Self-assembling Capabilities. Technical report, ETH-Zürich, 2002.
- [254] MongoDB. <https://www.mongodb.com/>. [Online; accessed 8-August-2024].
- [255] Roberto Montemanni, Luca Maria Gambardella, Andrea Emilio Rizzoli, and Alberto V Donati. Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization*, 10(4):327–343, 2005.
- [256] Honda Motor. Safe swarm. https://global.honda/innovation/CES/2019/safe_swarm.html, 2019. [Online; accessed 01-October-2019].
- [257] Jean-Baptiste Mouret and Konstantinos Chatzilygeroudis. 20 years of reality gap: a few thoughts about simulators in evolutionary robotics. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1121–1124. ACM, 2017. DOI: 10.1145/3067695.3082052.
- [258] Mehdi Moussaïd, Dirk Helbing, Simon Garnier, Anders Johansson, Maud Combe, and Guy Theraulaz. Experimental study of the behavioural mechanisms underlying self-organization in human crowds. *Proceedings of the Royal Society of London B: Biological Sciences*, 276(1668):2755–2762, 2009.
- [259] Ahmad Mozaffari, Alireza Fathi, and Saeed Behzadipour. The great salmon run: A novel bio-inspired algorithm for artificial system design and optimisation. *International Journal of Bio-Inspired Computation*, 4(5):286–301, 2012.
- [260] Rolex Muceka, Tonny Kukeera, Yunus Alokore, Kebir Noara, and Sebastian Groh. Integrating a Solar PV System with a Household Based Backup Generator for Hybrid Swarm Electrification: A Case Study of Nigeria. In *Africa-EU Renewable Energy Research and Innovation Symposium 2018 (RERIS 2018)*, pages 43–58. Springer, Cham, 2018.

- [261] Satoshi Murata, Eiichi Yoshida, Akiya Kamimura, Haruhisa Kurokawa, Kohji Tomita, and Shigeru Kokaji. M-TRAN: Self-reconfigurable Modular Robotic System. *IEEE/ASME Transactions on Mechatronics*, 7(4):431–441, 2002.
- [262] Robin R. Murphy, Satoshi Tadokoro, Daniele Nardi, Adam Jacoff, Paolo Fiorini, Howie Choset, and Aydan M. Erkmén. Search and rescue robotics. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, pages 1151–1173. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [263] Smiljana Mutic, Yvonne F. Br  nner, Rea Rodriguez-Raecke, Martin Wiesmann, and Jessica Freiherr. Chemosensory danger detection in the human brain: Body odor communicating aggression modulates limbic system activation. *Neuropsychologia*, 99(September 2016):187–198, 2017.
- [264] Seongin Na, Hanlin Niu, Barry Lennox, and Farshad Arvin. Universal artificial pheromone framework with deep reinforcement learning for robotic systems. In *6th International Conference on Control and Robotics Engineering (ICCRE)*, pages 28–32. IEEE, 2021.
- [265] Seongin Na, Hanlin Niu, Barry Lennox, and Farshad Arvin. Bio-inspired collision avoidance in swarm systems via deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 2022.
- [266] Seongin Na, Yiping Qiu, Ali E Turgut, Ji  r   Ulrich, Tom     Krajn  k, Shigang Yue, Barry Lennox, and Farshad Arvin. Bio-inspired artificial pheromone system for swarm robotics applications. *Adaptive Behavior*, 29(4):395–415, 2021.
- [267] Seongin Na, Mohsen Raoufi, Ali Emre Turgut, Tom     Krajn  k, and Farshad Arvin. Extended artificial pheromone system for swarm robotic applications. In *Artificial Life Conference Proceedings*, pages 608–615. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2019.
- [268] Seongin Na, Tom     Rou    ek, Ji  r   Ulrich, Jan Pikman, Tom     Krajn  k, Barry Lennox, and Farshad Arvin. Federated reinforcement learning for collective navigation of robotic swarms. *IEEE Transactions on Cognitive and Developmental Systems*, 2023.
- [269] Jawad Nagi. *Symbiotic interaction between humans and robot swarms*. PhD thesis, Department of Informatics, University of Lugano, Switzerland, 2016.
- [270] Jawad Nagi, Alessandro Giusti, Luca Gambardella, and Gianni A. Di Caro. Human-swarm interaction using spatial gestures. In *Proceedings of the 27th IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3834–3841, 2014.

- [271] Changjoo Nam and Dylan A. Shell. Assignment algorithms for modeling resource contention in multirobot task allocation. *IEEE Transactions on Automation Science and Engineering*, 12(3):889–900, 2015.
- [272] Yadati Narahari, Dinesh Garg, Ramasuri Narayanam, and Hastagiri Prakash. *Game theoretic problems in network economics and mechanism design solutions*. Springer, 2009.
- [273] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.
- [274] Claus Ballegaard Nielsen, Peter Gorm Larsen, John Fitzgerald, Jim Woodcock, and Jan Peleska. Systems of systems engineering: Basic concepts, model-based techniques, and research directions. *ACM Computing Surveys*, 48(2), Sep 2015.
- [275] Martin A. Nowak, Corina E. Tarnita, and Edward O. Wilson. The evolution of eusociality. *Nature*, 466(7310):1057–1062, Aug 2010.
- [276] Olusegun Olorunda and Andries P. Engelbrecht. Measuring exploration/exploitation in particle swarms using swarm diversity. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1128–1134, 2008.
- [277] Nestor I. Ospina, Eduardo Mojica-Nava, Luis G. Jaimes, and Juan M. Calderón. Argrohbots: An affordable and replicable ground homogeneous robot swarm testbed. *IFAC-PapersOnLine*, 54(13):256–261, 2021.
- [278] Christoph Osterloh, Thilo Pionteck, and Erik Maehle. MONSUN II: A small and Inexpensive AUV for Underwater Swarms. In *Proceedings of the 7th German Conference on Robotics*, pages 1–6, 2012.
- [279] Raymond Oung. The Distributed Flight Array: Summary. <https://www.youtube.com/watch?v=fcradVE9uts>, 2013. [Online; accessed 30-January-2020].
- [280] Raymond Oung and Raffaello D’Andrea. The Distributed Flight Array. *Mechatronics*, 21(6):908–917, 2011.
- [281] Rafael S. Parpinelli and Heitor S. Lopes. New inspirations in swarm intelligence: A survey. *International Journal of Bio-Inspired Computation*, 3(1):1–16, 2011.
- [282] Rafael S Parpinelli and Heitor S Lopes. New inspirations in swarm intelligence: A survey. *International Journal of Bio-Inspired Computation*, 3(1):1–16, 2011.

- [283] Jayam Patel and Carlo Pinciroli. Improving human performance using mixed granularity of control in multi-human multi-robot interaction. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1135–1142. IEEE, 2020.
- [284] Jayam Patel, Yicong Xu, and Carlo Pinciroli. Mixed-granularity human-swarm interaction. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1059–1065. IEEE, 2019.
- [285] David Payton, Mike Daily, Regina Estowski, Mike Howard, and Craig Lee. Pheromone robotics. *Autonomous Robots*, 11(3):319–324, 2001.
- [286] Kirstin Hagelskjaer Petersen, Radhika Nagpal, and Justin K Werfel. Termes: An Autonomous Robotic System for Three-dimensional Collective Construction. *Robotics: science and systems VII*, 2011.
- [287] Daniel Pickem, Paul Glotfelter, Li Wang, Mark Mote, Aaron Ames, Eric Feron, and Magnus Egerstedt. The robotarium: A remotely accessible swarm robotics research testbed. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1699–1706. IEEE, 2017.
- [288] Daniel Pickem, Myron Lee, and Magnus Egerstedt. The GRITSBot in its Natural Habitat – a Multi-robot Testbed. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4062–4067, 2015.
- [289] Carlo Pinciroli and Giovanni Beltrame. Swarm-oriented Programming of Distributed Robot Networks. *Computer*, 49(12):32–41, 2016.
- [290] Carlo Pinciroli, Vito Trianni, Rehan O’Grady, Giovanni Pini, Arne Bratsch, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, et al. Argos: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm intelligence*, 6(4):271–295, 2012.
- [291] G. Porter. Models of Proba-3 designs. http://www.esa.int/spaceinimages/Images/2016/05/Models_of_Proba-3_designs, 2016. [Online; accessed 9-July-2019].
- [292] Christian Prehofer and Christian Bettstetter. Self-organization in communication networks: principles and design paradigms. *IEEE Communications Magazine*, 43(7):78–85, 2005.
- [293] James A. Preiss, Wolfgang Hönig, Gaurav S. Sukhatme, and Nora Ayanian. CrazySwarm: A large Nano-quadcopter Swarm. In *Proceedings of the International Conference on Robotics and Automation*, pages 3299–3304, 2017. Software available at <https://github.com/USC-ACTLab/crazyswarm>.

- [294] George Preti, Charles J. Wysocki, Kurt T. Barnhart, Steven J. Sondheimer, and James J. Leyden. Male Axillary Extracts Contain Pheromones that Affect Pulsatile Secretion of Luteinizing Hormone and Mood in Women Recipients¹. *Biology of Reproduction*, 68(6):2107–2113, 06 2003.
- [295] CPSwarm H2020 project. CPSwarm Modeler. <https://github.com/cpswarm/modelio-cpswarm-modeler>, 2019. [Online; accessed 23-March-2023].
- [296] CPSwarm H2020 project. CPSwarm Code Generator. <https://github.com/cpswarm/code-generator>, 2020. [Online; accessed 23-March-2023].
- [297] CPSwarm H2020 project. CPSwarm Communication Library. <https://github.com/cpswarm/swarmio>, 2022. [Online; accessed 23-March-2023].
- [298] Przemyslaw Prusinkiewicz, Mitra Shirmohammadi, and Faramarz Samavati. L-systems in geometric modeling. *International Journal of Foundations of Computer Science*, 23(01):133–146, 2012.
- [299] Amanda Purington, Jessie G Taft, Shruti Sannon, Natalya N Bazarova, and Samuel Hardman Taylor. Alexa is my new bff: Social roles, user satisfaction, and personification of the amazon echo. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 2853–2859, 2017.
- [300] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source robot operating system. In *Proceedings of the ICRA Workshop on Open Source Software in Robotics*, May 2009.
- [301] Deepak Rai and Kirti Tyagi. Bio-inspired optimization techniques: a critical comparative study. *ACM SIGSOFT Software Engineering Notes*, 38(4):1–7, 2013.
- [302] Steven Railsback, Daniel Ayllón, Uta Berger, Volker Grimm, Steven Lytinen, Colin Sheppard, and Jan Christoph Thiele. Improving execution speed of models implemented in netlogo. *Journal of Artificial Societies and Social Simulation*, 2017.
- [303] Steven F Railsback and Volker Grimm. *Agent-based and individual-based modeling: a practical introduction*. Princeton university press, "2nd" edition, 2019.
- [304] Bhuvaneswari Ramachandran, Sanjeev K. Srivastava, Chris S. Edrington, and David A. Cartes. An intelligent auction scheme for smart grid market using a hybrid immune algorithm. *IEEE Transactions on Industrial Electronics*, 58(10):4603–4612, 2011.

- [305] Mohsen Raoufi, Ali Emre Turgut, and Farshad Arvin. Self-organized collective motion with a simulated real robot swarm. In *Proceedings of the Annual Conference Towards Autonomous Robotic Systems*, pages 263–274. Springer, 2019.
- [306] Martin Reed, Aliko Yiannakou, and Roxanne Evering. An ant colony algorithm for the multi-compartment vehicle routing problem. *Applied Soft Computing*, 15:169–176, 2014.
- [307] Andreagioanni Reina, Alexander Cope, Eleftherios Nikolaidis, James Marshall, and Chelsea Sabo. Ark: Augmented reality for kilobots. *IEEE Robotics and Automation Letters*, PP:1–1, 05 2017.
- [308] Fatemeh Rekabi, Farzad A Shirazi, Mohammad Jafar Sadigh, and Mahmood Saadat. Distributed output feedback nonlinear h_∞ formation control algorithm for heterogeneous aerial robotic teams. *Robotics and Autonomous Systems*, 136:103689, 2021.
- [309] Fatemeh Rekabi-Bana, Junyan Hu, Tomáš Krajník, and Farshad Arvin. Unified robust path planning and optimal trajectory generation for efficient 3d area coverage of quadrotor uavs. *IEEE Transactions on Intelligent Transportation Systems*, 2023.
- [310] Intel Corporation. Intel drone light show breaks guinness world records title at olympic winter games pyeongchang. <https://newsroom.intel.com/news-releases/intel-drone-light-show-breaks-guinness-world-records-title-olympic-winter-games-pyeongchang-2018/> 2018. [Online; accessed 30-September-2019].
- [311] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, 1987.
- [312] Fanny Riedo, Morgane Chevalier, Stéphane Magnenat, and Francesco Mondada. Thymio II, a Robot that Grows Wiser with Children. In *Proceedings of the IEEE Workshop on Advanced Robotics and its Social Impacts*, pages 187–193, 2013.
- [313] James F. Roberts, Timothy S. Stirling, Jean-Christophe Zufferey, and Dario Floreano. Quadrotor Using Minimal Sensing For Autonomous Indoor Flight. In *Proceedings of the European Micro Air Vehicle Conference and Flight Competition*, 2007.
- [314] Rui P Rocha, David Portugal, Micael Couceiro, Filipe Araújo, Paulo Menezes, and Jorge Lobo. The CHOPIN project: Cooperation between human and robotic teams in catastrophic incidents. In *Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 1–4, 2013.

- [315] Eric Rohmer, Surya PN Singh, and Marc Freese. V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326. IEEE, 2013.
- [316] Michael Rubenstein, Christian Ahler, Nick Hoff, Adrian Cabrera, and Radhika Nagpal. Kilobot: A low cost Robot with Scalable Operations Designed for Collective Behaviors. *Robotics and Autonomous Systems (Elsevier)*, 62(7):966–975, 2014.
- [317] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *2012 IEEE International Conference on Robotics and Automation*, pages 3293–3298, May 2012.
- [318] Ray A. Russell. Ant trails - an example for robots to follow? In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 4, pages 2698–2703 vol.4, May 1999.
- [319] Joan Saez-Pons, Lyuba Alboul, Jacques Penders, and Leo Nomdedeu. Multi-robot team formation control in the GUARDIANS project. *Industrial Robot: An International Journal*, 37(4):372–383, 2010.
- [320] Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *Proceedings of the International Workshop on Swarm Robotics*, pages 10–20, 2004.
- [321] Muhammad Salman, David Garzón Ramos, Ken Hasselmann, and Mauro Birattari. Phormica: Photochromic pheromone release and detection system for stigmergic coordination in robot swarms. *Frontiers in Robotics and AI*, 7:195, 2020.
- [322] William H. Sandholm. *Population Games and Evolutionary Dynamics*. MIT Press, 2010.
- [323] Hiroki Sayama. Morphologies of self-organizing swarms in 3d swarm chemistry. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, pages 577–584. ACM, 2012.
- [324] Hiroki Sayama. PyCX: A python-based simulation code repository for complex systems education. *Complex Adaptive Systems Modeling*, 1(1):1–10, 2013.
- [325] Jürgen Scherer, Saeed Yahyanejad, Samira Hayat, Evsen Yanmaz, Torsten Andre, Asif Khan, Vladimir Vukadinovic, Christian Bettstetter, Hermann Hellwagner, and Bernhard Rinner. An autonomous multi-uav system for search and rescue. In *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*, pages 33–38, 2015.

- [326] Thomas Schmickl, Stjepan Bogdan, Luís Correia, Serge Kernbach, Francesco Mondada, Michael Bodi, Alexey Gribovskiy, Sibylle Hahshold, Damjan Miklic, Martina Szopek, et al. Assisi: Mixing animals with robots in a hybrid society. In *Proceedings of the Conference on Biomimetic and Biohybrid Systems*, pages 441–443, 2013.
- [327] Thomas Schmickl and Heiko Hamann. Beeclust: A swarm algorithm derived from honeybees. In Yang Xiao, editor, *Bio-inspired Computing and Networking*, pages 95–137. CRC Press, 2011.
- [328] Thomas Schmickl, Martin Stefanec, and Karl Crailsheim. How a life-like system emerges from a simplistic particle motion law. *Scientific reports*, 6(1):1–15, 2016.
- [329] Thomas Schmickl, Ronald Thenius, Christoph Möslinger, Gerald Radspieler, Serge Kernbach, and Karl Crailsheim. Get in touch: Cooperative decision making based on robot-to-robot collisions. *Autonomous Agents and Multi-Agent Systems*, 18(1):133–155, 2008.
- [330] Thomas Schmickl, Ronald Thenius, Christoph Moslinger, Jon Timmis, Andy Tyrrell, Mark Read, James Hilder, Jose Halloy, Alexandre Campo, Cesare Stefanini, et al. Cocoro - the self-aware underwater swarm. In *Proceedings of the 5th IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops*, pages 120–126, 2011.
- [331] Thomas Schmickl, Franz Wotawa, Ronald Thenius, and Joshua Cherian Varughese. Fstaxis algorithm: Bio-inspired emergent gradient taxis. In *Proceedings of the Artificial Life Conference 2016 13*, pages 330–337. MIT Press, 2016.
- [332] Melanie Schranz, Alessandra Bagnato, Etienne Brosse, and Wilfried Elmenreich. Modelling a CPS swarm system: A simple case study. In *Proceedings of the International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 615–624. SciTePress, January 2018.
- [333] Melanie Schranz, Gianni A Di Caro, Thomas Schmickl, Wilfried Elmenreich, Farshad Arvin, Ahmet Şekercioğlu, and Micha Sende. Swarm intelligence and cyber-physical systems: Concepts, challenges and future trends. *Swarm and Evolutionary Computation*, 60:100762, 2020.
- [334] Melanie Schranz, Kseniia Harshina, Peter Forgacs, and Fred Buining. Agent-based modeling in the edge continuum using swarm intelligence. In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence*. SciTePress, 2024.
- [335] Melanie Schranz, Micha Sende, Alessandra Bagnato, and Etienne Brosse. Modeling swarm intelligence algorithms for cps swarms. *Ada User Journal*, 40(3):169–177, September 2019.

- [336] Melanie Schranz, Martina Umlauft, and Wilfried Elmenreich. Bottom-up job shop scheduling with swarm intelligence in large production plants. In *Proceedings of the 11th International Conference on Simulation and Modeling Methodologies, Technologies and Applications - SIMULTECH*. SciTePress, July 2021.
- [337] Melanie Schranz, Martina Umlauft, Micha Sende, and Wilfried Elmenreich. Swarm robotic behaviors and current applications. *Frontiers in Robotics and AI*, 7:36, 2020.
- [338] Adam M. Schroeder and Manish Kumar. Design of decentralized chemotactic control law for area coverage using swarm of mobile robots. In *Proceedings of the American Control Conference*, pages 4317–4322, 2016.
- [339] Israel’s Homeland Security. Fish swarm – model for energy-saving autonomous vehicle swarm. <https://i-hls.com/archives/84280>, 2018. [Online; accessed 30-September-2019].
- [340] Madhubhashi Senanayake, Ilankaikone Senthoooran, Jan Carlo Barca, Hoam Chung, Joarder Kamruzzaman, and Manzur Murshed. Search and tracking algorithms for swarms of robots: A survey. *Robotics and Autonomous Systems*, 75:422–434, 2016.
- [341] Micha Sende. swarm_behaviors. https://wiki.ros.org/swarm_behaviors, 2019. [Online; accessed 23-March-2023].
- [342] Micha Sende. swarm_functions. https://wiki.ros.org/swarm_functions, 2019. [Online; accessed 23-March-2023].
- [343] Micha Sende. CPSwarm Hardware Functions. https://github.com/cpswarm/hardware_functions, 2023. [Online; accessed 23-March-2023].
- [344] Micha Sende. Sensing and Actuation. https://github.com/cpswarm/sensing_actuation, 2023. [Online; accessed 23-March-2023].
- [345] Micha Sende, Melanie Schranz, , Gianluca Prato, Etienne Brosse, Omar Morando, and Martina Umlauft. Engineering swarms of cyber-physical systems with the cpswarm workbench. *Journal of Intelligent & Robotic Systems*, 102(83):18, 2021.
- [346] Sebastian Senge and Horst F Wedde. Bee-inspired road traffic control as an example of swarm intelligence in cyber-physical systems. In *Proceedings of the 38th Euromicro Conference on Software Engineering and Advanced Applications*, pages 258–265. IEEE, 2012.
- [347] Jörg Seyfried, Marc Szymanski, Natalie Bender, Ramon Estaña, Michael Thiel, and Heinz Wörn. The I-SWARM Project: Intelligent Small World Autonomous Robots for Micro-manipulation. In *Proceedings of the International Workshop on Swarm Robotics*, pages 70–83, 2004.

- [348] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017.
- [349] Hamed Shah-Hosseini. Principal components analysis by the galaxy-based search algorithm: A novel metaheuristic for continuous optimisation. *International Journal of Computational Science and Engineering*, 6(1-2):132–140, 2011.
- [350] Adi Shklarsh, Gil Ariel, Elad Schneidman, and Eshel Ben-Jacob. Smart swarms of bacteria-inspired agents with performance adaptable interactions. *PLoS Computational Biology*, 7(9):1–11, 2011.
- [351] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.
- [352] Ola Shorinwa, Javier Yu, Trevor Halsted, Alex Koufos, and Mac Schwager. Distributed multi-target tracking for autonomous vehicle fleets. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3495–3501. IEEE, 2020.
- [353] Abdul A Siddiqi and Simon M Lucas. A comparison of matrix rewriting versus direct encoding for evolving neural networks. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360)*, pages 392–397. IEEE, 1998.
- [354] Jennifer Simonjan, Stefano Ricardo Probst, and Melanie Schranz. Inducing defenders to mislead an attacking uav swarm. In *Proceedings of the 42nd IEEE International Conference on Distributed Computing Systems Workshops*, pages 278–283, 2022.
- [355] Jorge M Soares, Inaki Navarro, and Alcherio Martinoli. The Khepera IV Mobile Robot: Performance Evaluation, Sensory Data and Software Toolbox. In *Proceedings of the 2nd Iberian Robotics Conference*, pages 767–781, 2016.
- [356] Anita Sobe. *Self-organizing Multimedia Delivery*. Phd thesis, Alpen-Adria Universität Klagenfurt, 2012.
- [357] Anita Sobe and Wilfried Elmenreich. Smart microgrids: Overview and outlook. In *Proceedings of the ITG INFORMATIK Workshop on Smart Grids*, Braunschweig, Germany, September 2012.
- [358] Anita Sobe, Wilfried Elmenreich, Tibor Szkaliczki, and Laszlo Böszörményi. SEAHORSE: Generalizing an artificial hormone system algorithm to a middleware for search and delivery of information units. *Computer Networks*, 80:124–142, 2015.

- [359] Anita Sobe, Istvan Fehérvári, and Wilfried Elmenreich. FREVO: A tool for evolving and evaluating self-organizing systems. In *Proceedings of the 1st International Workshop on Evaluation for Self-Adaptive and Self-Organizing Systems*, pages 105–110, 2012.
- [360] Ricard V. Solé, Susanna C. Manrubia, Bartolo Luque, Jordi Delgado, and Jordi Bascompte. Phase transitions and complex systems: Simple, nonlinear models capture complex systems at the edge of chaos. *Complexity*, 1(4):13–26, 1996.
- [361] Kian Lun Soon, Joanne Mun-Yee Lim, and Rajendran Parthiban. Coordinated traffic light control in cooperative green vehicle routing for pheromone-based multi-agent systems. *Applied Soft Computing*, 81:105486, 2019.
- [362] Kenneth Stanley. rtNEAT C++. <https://nn.cs.utexas.edu/?rtNEAT>, 2006. [Online; accessed 23-March-2023].
- [363] Kenneth O. Stanley and Risto Miikkulainen. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*, 10(2):99–127, 2002.
- [364] David Steber, Peter Bazan, and Reinhard German. Swarm - increasing households’ internal pv consumption and offering primary control power with distributed batteries. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9424, pages 3–11. Springer Verlag, 2015.
- [365] Nathalie Steinhauer, Kelly Kulhanek, Karina Antúnez, Hannelie Human, Panuwan Chantawannakul, Marie-Pierre Chauzat, and Dennis van Engelsdorp. Drivers of colony losses. *Current Opinion in Insect Science*, 26:142 – 148, 2018.
- [366] John D Stermann. System dynamics modeling: tools for learning in a complex world. *California management review*, 43(4):8–25, 2001.
- [367] Apache Storm. <https://storm.apache.org/>. [Online; accessed 8-August-2024].
- [368] Daniel P Stormont. Robot swarms for planetary exploration. In *Proceedings of the 4th International Conference and Exposition on Robotics for Challenging Situations and Environments*, pages 347–352, 2000.
- [369] Steven H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Studies in Nonlinearity. Avalon Publishing, 2014.

- [370] Daniel Strömbom, Mattias Siljestam, Jinha Park, and David JT Sumpter. The shape and dynamics of local attraction. *The European Physical Journal Special Topics*, 224(17):3311–3323, 2015.
- [371] Gerald Jay Sussman. Building robust systems an essay. *Massachusetts Institute of Technology*, 2007.
- [372] Project SWARMIX: Synergistic interactions in swarms of heterogeneous agents (2011-2014). <http://www.swarmix.org/>. [Online; accessed 14-August-2019].
- [373] S. Swetha, R. Anandan, and K. Kalaivani. An Investigation of Micro Aerial Vehicles (μ AV). *International Journal of Engineering & Technology*, 7(2.31):174–177, 2018.
- [374] Martina Szopek, Thomas Schmickl, Ronald Thenius, Gerald Radspieler, and Karl Crailsheim. Dynamics of collective decision making of honeybees in complex temperature fields. *PloS one*, 8(10):e76250, 2013.
- [375] Kenichi Tamura and Keiichiro Yasuda. Spiral dynamics inspired optimization. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 15(8):1116–1122, 2011.
- [376] Ying Tan and Yuanchun Zhu. Fireworks algorithm for optimization. In *Proceedings of the International Conference on Swarm Intelligence*, pages 355–364. Springer, 2010.
- [377] Arash Tavakoli, Haig Nalbandian, and Nora Ayanian. Crowdsourced coordination through online games. In *Proceedings of the 11th ACM/IEEE International Conference on Human-Robot Interaction*, pages 527–528, 2016.
- [378] TensorFlow. <https://www.tensorflow.org/>. [Online; accessed 8-August-2024].
- [379] Erich C. Teppan. Dispatching rules revisited – a large scale job shop scheduling experiment. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pages 561–568, 2018.
- [380] Ronald Thenius, Daniel Moser, Joshua Cherian Varughese, Serge Kernbach, Igor Kuksin, Olga Kernbach, Elena Kuksina, Nikola Mišković, Stjepan Bogdan, Tamara Petrović, et al. subCULTron-Cultural Development as a Tool in Underwater Robotics. In *Proceedings of the Artificial Life and Intelligent Agents Symposium*, pages 27–41. Springer, 2016.
- [381] Jan C Thiele. R marries NetLogo: introduction to the RNetLogo package. *Journal of Statistical Software*, 58:1–41, 2014.

- [382] Seth Tisue and Uri Wilensky. Netlogo: A simple environment for modeling complexity. In *International conference on complex systems*, volume 21, pages 16–21. Boston, MA, 2004.
- [383] Ioan C. Trelea. The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters*, 85(6):317–325, 2003.
- [384] Elio Tuci and Alexandre Rabérin. On the design of generalist strategies for swarms of simulated robots engaged in a task-allocation scenario. *Swarm Intelligence*, 9(4):267–290, 2015.
- [385] Ali E Turgut, Hande Çelikkanat, Fatih Gökçe, and Erol Şahin. Self-organized flocking in mobile robot swarms. *Swarm Intelligence*, 2:97–120, 2008.
- [386] Ali Emre Turgut, İhsan Caner Boz, İlkin Ege Okay, Eliseo Ferrante, and Cristián Huepe. Interaction network effects on position-and velocity-based models of collective motion. *Journal of the Royal Society Interface*, 17(169):20200165, 2020.
- [387] Alan M. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641):37–72, 1952.
- [388] Fritz Ulbrich, Simon Sebastian Rotter, and Raul Rojas. Adapting to the traffic swarm: Swarm behaviour for autonomous cars. In Ying Tan, editor, *Handbook of Research on Design, Control, and Modeling of Swarm Robotics*, chapter 10, pages 263–285. IGI Global, 2016.
- [389] Martina Umlauf. Swarm Fab Simulation. <https://swarmfabsim.github.io>, 2023. [Online; accessed 23-March-2023].
- [390] Martina Umlauf, Melanie Schranz, and Wilfried Elmenreich. Swarm-FabSim: A simulation framework for bottom-up optimization in flexible job-shop scheduling using Netlogo. In *Proceedings of the 12th International Conference on Simulation and Modeling Methodologies, Technologies and Applications - SIMULTECH*, pages 271–279. SciTePress, July 2022.
- [391] Gabriele Valentini, Anthony Antoun, Marco Trabattoni, Bernát Wiandt, Yasumasa Tamura, Etienne Hocquard, Vito Trianni, and Marco Dorigo. Kilogrid: a novel experimental environment for the kilobot robot. *Swarm Intelligence*, 12(3):245–266, Sep 2018.
- [392] Richard Vaughan. Massively multi-robot simulation in stage. *Swarm intelligence*, 2(2-4):189–208, 2008.

- [393] Ganesh K Venayagamoorthy. Potentials and promises of computational intelligence for smart grids. In *2009 IEEE Power & Energy Society General Meeting*, pages 1–6. IEEE, 2009.
- [394] Tamás Vicsek, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 75(6):1226–1229, 1995.
- [395] Tamás Vicsek and Anna Zafeiris. Collective motion. *Physics reports*, 517(3-4):71–140, 2012.
- [396] 4D Virtualiz. <https://www.4d-virtualiz.com/en/>, 2024. [Online; accessed 23-March-2023].
- [397] Alessandra Vitanza, Paolo Rossetti, Francesco Mondada, and Vito Trianni. Robot Swarms as an Educational Tool: The Thymio’s Way. *International Journal of Advanced Robotic Systems*, 16(1), 2019.
- [398] Richard G Vogt, Lynn M Riddiford, and Glenn D Prestwich. Kinetic properties of a sex pheromone-degrading enzyme: the sensillar esterase of *antheraea polyphemus*. *Proceedings of the National Academy of Sciences*, 82(24):8827–8831, 1985.
- [399] Mostafa Wahby, Mary Katherine Heinrich, Daniel Nicolas Hofstadler, Ewald Neufeld, Igor Kuksin, Payam Zahadat, Thomas Schmickl, Phil Ayres, and Heiko Hamann. Autonomously shaping natural climbing plants: A bio-hybrid approach. *Open Science*, 5(10):180296, 2018.
- [400] Pangwei Wang, Hui Deng, Juan Zhang, Li Wang, Mingfang Zhang, and Yongfu Li. Model predictive control for connected vehicle platoon under switching communication topology. *IEEE Transactions on Intelligent Transportation Systems*, 2021.
- [401] Shiyi Wang, Ali E Turgut, Thomas Schmickl, Barry Lennox, and Farshad Arvin. Investigation of cue-based aggregation behaviour in complex environments. In *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 18–36. Springer, 2020.
- [402] Nicholas W. Watkins, Gunnar Pruessner, Sandra C. Chapman, Norma B. Crosby, and Henrik J. Jensen. 25 years of self-organized criticality: Concepts and controversies. *Space Science Reviews*, 198(1):3–44, 2016.
- [403] Hongxing Wei, Yingpeng Cai, Haiyuan Li, Dezhong Li, and Tianmiao Wang. Sambot: A Self-assembly Modular Robot for Swarm Robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 66–71. IEEE, 2010.

- [404] Uri Wilensky. Netlogo, 1999. <http://ccl.northwestern.edu/netlogo/>.
- [405] Sean Wilson, Aurélie Buffin, Stephen C Pratt, and Spring Berman. Multi-robot Replication of Ant Collective Towing Behaviours. *Royal Society open science*, 5(10):180409, 2018.
- [406] Sean Wilson, Ruben Gameros, Michael Sheely, Matthew Lin, Kathryn Dover, Robert Gevorkyan, Matt Haberland, Andrea Bertozzi, and Spring Berman. Pheeno, a Versatile Swarm robotic Research and Education Platform. *IEEE Robotics and Automation Letters*, 1(2):884–891, 2016.
- [407] Wayne H. Wolf. Hardware-software co-design of embedded systems. *Proceedings of the IEEE*, 82(7):967–989, 1994.
- [408] Kefan Wu, Junyan Hu, Zhengtao Ding, and Farshad Arvin. Distributed bearing-only formation control for heterogeneous nonlinear multi-robot systems. *IFAC-PapersOnLine*, 56(2):3447–3452, 2023.
- [409] Kefan Wu, Junyan Hu, Zhengtao Ding, and Farshad Arvin. Finite-time fault-tolerant formation control for distributed multi-vehicle networks with bearing measurements. *IEEE Transactions on Automation Science and Engineering*, 2023.
- [410] Kefan Wu, Junyan Hu, Zhenhong Li, Zhengtao Ding, and Farshad Arvin. Distributed collision-free bearing coordination of multi-uav systems with actuator faults and time delays. *IEEE Transactions on Intelligent Transportation Systems*, 2024.
- [411] Tristram D. Wyatt. *Pheromones and Animal Behavior: Chemical Signals and Signatures*. Cambridge University Press, 2014.
- [412] Wyss Institute. Programmable Robot Swarms. <https://wyss.harvard.edu/technology/programmable-robot-swarms/>, 2017. [Online; accessed 06-November-2019].
- [413] Songtao Xie, Junyan Hu, Parijat Bhowmick, Zhengtao Ding, and Farshad Arvin. Distributed motion planning for safe autonomous vehicle overtaking via artificial potential field. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):21531–21547, 2022.
- [414] Songtao Xie, Junyan Hu, Zhengtao Ding, and Farshad Arvin. Collaborative overtaking of multi-vehicle systems in dynamic environments: A distributed artificial potential field approach. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 873–878. IEEE, 2021.
- [415] Songtao Xie, Junyan Hu, Zhengtao Ding, and Farshad Arvin. Cooperative adaptive cruise control for connected autonomous vehicles using spring damping energy model. *IEEE Transactions on Vehicular Technology*, 2022.

- [416] Xin-She Yang. Flower pollination algorithm for global optimization. In *Proceedings of the International Conference on Unconventional Computing and Natural Computation*, pages 240–249, 2012.
- [417] Xin-She Yang, Zhihua Cui, Renbin Xiao, Amir Hossein Gandomi, and Mehmet Karamanoglu. *Swarm Intelligence and Bio-inspired Computation: Theory and Applications*. Elsevier, 2013.
- [418] Xin-She Yang, Suash Deb, Yu-Xin Zhao, Simon Fong, and Xingshi He. Swarm intelligence: Past, present and future. *Soft Computing*, pages 1–11, 2017.
- [419] Shoji Yano. Kajima to develop automated construction machinery for building on Mars, Moon. <https://asia.nikkei.com/Tech-Science/Tech/Kajima-to-develop-automated-construction-machinery-for-building-on-Mars-moon>, 2016. [Online; accessed 9-July-2019].
- [420] Sha Yi, Zeynep Temel, and Katia Sycara. Puzzlebots: Physical coupling of robot swarms. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8742–8748, 2021.
- [421] Bin Yu, Zhong-Zhen Yang, and Baozhen Yao. An improved ant colony optimization for vehicle routing problem. *European Journal of Operational Research*, 196(1):171–176, 2009.
- [422] Zhi Yuan, Marco A. Montes de Oca, Mauro Birattari, and Thomas Stützle. Continuous optimization algorithms for tuning real and integer parameters of swarm intelligence algorithms. *Swarm Intelligence*, 6(1):49–75, 2012.
- [423] Xizhe Zang, Sajid Iqbal, Yanhe Zhu, Xinyu Liu, and Jie Zhao. Applications of chaotic dynamics in robotics. *International Journal of Advanced Robotic Systems*, 13(2), 60 p., 2016.
- [424] Fangbo Zhang, Andrea L. Bertozzi, Karthik Elamvazhuthi, and Spring Berman. Performance bounds on spatial coverage tasks by stochastic robotic swarms. *IEEE Transactions on Automatic Control*, 63(6):1473–1488, 2018.
- [425] Guohui Zhang, Xinyu Shao, Peigen Li, and Liang Gao. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 56(4):1309–1318, 2009.
- [426] Yuan Zhang, Yu Yuan, and Kejing Lu. E-commerce information system data analytics by advanced aco for asymmetric capacitated vehicle delivery routing. *Information Systems and e-Business Management*, pages 1–19, 2019.

- [427] Yating Zheng, Cristián Huepe, and Zhangang Han. Experimental capabilities and limitations of a position-based control algorithm for swarm robotics. *Adaptive Behavior*, 30(1):19–35, 2022.
- [428] George R. Zug. Walk pattern analysis of cryptodiran turtle gaits. *Animal Behaviour*, 20(3):439–443, 1972.
- [429] Victor Zykov, Efstathios Mytilinaios, Mark Desnoyer, and Hod Lipson. Evolved and Designed Self-Reproducing Modular Robotics. *IEEE Transactions on Robotics*, 23(2):308–319, 2007.

Index

A

Actuators 4, 8, 27, 28, 30–32, 43, 59, 60, 63, 64, 85, 97, 103, 120, 128
Adaptability 3, 7, 44, 45, 54, 75, 109, 126
Agent-based models 41, 65

B

BeeGround 68, 81–83

C

Collective motion 46, 47, 49, 70, 117
Cyber-physical systems 1, 8, 20, 54, 65, 67

D

Distributed control 8, 109, 113

E

Emergence 2, 6, 8, 11, 12, 109, 113, 126
Emergent behavior 6, 10, 24, 31, 41, 118, 121, 124
Evolutionary algorithms 61
Evolutionary optimization 31, 57, 58, 65
Exploration vs. Exploitation 10, 18, 19, 29, 44, 45, 50, 51, 66, 68, 72, 76, 79, 85, 99–101, 115, 123, 127

F

Feedback loops 12, 60
FREVO 58–65, 111

M

Micro-macro dynamics 12
Modular design 18
Multi-agent systems 6, 7, 60, 65, 77, 113

N

NetLogo 56, 68, 71, 72, 75

P

Pheromone communication 14, 82, 88, 89
Physics-based simulation 68, 70, 75–80, 82

R

Real-World deployment 112

S

Scalability 2, 3, 7, 11–13, 18, 27, 54, 93, 109, 113, 115, 116, 126, 127
Sensors 3, 4, 8, 9, 27–29, 31, 32, 43, 56, 58–60, 63, 64, 69, 76, 81, 84–86, 88, 89, 94–100, 103, 105, 106, 112, 118–121, 126, 128
Swarm behavior 1–3, 5, 10, 12, 20, 27–31, 33, 35–37, 43–45, 51, 69, 70, 75, 82, 92, 94, 101, 110, 119, 128
Swarm challenges 8, 12, 13, 22, 24, 38, 43, 46, 56, 65, 93, 100, 108–114, 117, 118, 120, 123, 124, 128

Swarm intelligence 1–10, 12, 38, 41, 95,
109–119, 122, 125–127
Swarm modeling 10, 11, 20, 22, 27, 28, 43,
71, 72, 81, 109, 113, 118, 127
Swarm robotics 2, 8, 13, 46, 47, 65–68,
75–77, 82, 85, 86, 88, 89, 94, 96, 99,
101, 103, 112, 113

Swarm simulation 65–67, 70, 81
System abstraction 20

T

Terrestrial robots 94