Programming and Compiler Toolchain for Multi-Agent Systems

Isara Tillekeratne
Department of Computer Engineering
University of Peradeniya
Peradeniya, Sri Lanka
isara.tillek@gmail.com

Kavinaya Yogendren
Department of Computer Engineering
University of Peradeniya
Peradeniya, Sri Lanka
kavinaya1212@gmail.com

Hashini Wijerathne
Department of Computer Engineering
University of Peradeniya
Peradeniya, Sri Lanka
hashini.sharintha@gmail.com

Isuru Nawinne
Department of Computer Engineering
University of Peradeniya
Peradeniya, Sri Lanka
isurunawinne@eng.pdn.ac.lk

Mahanama Wickramasinghe
Department of Computer Engineering
University of Peradeniya
Peradeniya, Sri Lanka
mahanamaw@eng.pdn.ac.lk

Roshan Ragel
Department of Computer Engineering
University of Peradeniya
Peradeniya, Sri Lanka
roshanr@eng.pdn.ac.lk

Sithumini Ekanayake
Department of Computer Engineering
University of Peradeniya
Peradeniya, Sri Lanka
sithuminie@eng.pdn.ac.lk

I. Introduction

Multi-agent systems (MAS) are a core area of research that refers to systems composed of multiple independent entities, often called agents, interacting and cooperating to achieve common goals. MAS research spans a range of technical problems, such as designing MAS to incentivise certain agent behaviours, developing algorithms to achieve specified goals in a MAS, and how information is communicated and propagated among agents. Concepts of multi-agent systems can be applied to enhance coordination, decision-making, and overall system performance in various domains, such as distributed sensor networks, Internet of Things (IoT) networks, autonomous vehicle systems, and online social networks comprising hardware and software-based agents. Out of these domains, our research focuses on swarm robotics which plays an important role within multi-agent systems.

Swarm robotics is a field of research and development that draws inspiration from the collective behaviour observed in natural swarms, such as flocks of birds, schools of fish, or colonies of ants. It involves the study of systems composed of multiple autonomous robots, known as agents, that work together to achieve common goals. Swarm robotics is based on the idea that simple individual agents, following simple rules, can collectively exhibit complex behaviours and accomplish tasks that would be challenging for a single robot to achieve. By coordinating their actions and interactions, the swarm of robots can achieve robustness, scalability, and adaptability. Research in swarm robotics focuses on developing swarm behavioural algorithms, strategies, and communication protocols, developing swarm programming frameworks, simulation platforms and hardware platforms.

When considering the characteristics of swarm robotics, the homogeneity of the agents, meaning that all the robots in the swarm have similar capabilities and functionalities, is crucial. This homogeneity allows for simplicity in the design and control of individual agents, as they can perform similar tasks and communicate with one another using the same set of rules. Being autonomous is another crucial characteristic of swarm robotics. Each agent can make decisions based on its local perception of the environment and interactions with neighbouring agents. This enables the swarm to exhibit self-organization and adaptability to changing conditions.

The principle of locality and decentralized behaviour is another fundamental in swarm robotics. It states that the behaviour of each agent depends primarily on its local interactions and the information it acquires from its immediate neighbours. Agents exchange information through local communication channels to make decisions and coordinate their actions. Decentralized behaviour implies that instead of relying on a central controller or leader, the behaviour of the swarm emerges from the interactions and decisions of the individual agents. This decentralized nature enhances robustness, as the failure or removal of a single agent does not significantly affect the overall performance of the swarm.

Some of the swarm behaviours that have been experimented with are aggregation, dispersion, pattern formation, collective movement, task allocation, source search and collective transportation of objects. These swarm behaviours can be

Identify applicable funding agency here. If none, delete this.

advantageous in real-world applications such as search and rescue missions, environmental monitoring, warehousing and logistics, construction, manufacturing processes, surveillance and security.

Achieving a complex swarm behaviour is not an easy task. Programming swarm robots can be tedious as it requires dealing with low-level complexities of handling and programming each robot and the interactions between the robots such that complex collective behaviours are achieved. Most of the available frameworks for swarm programming focus only on software-level simulations, which do not discuss extending them to real hardware robot platforms, and they are limited only to a few pre-programmed sets of behaviours and do not give developers the ability to change the inbuilt behaviours or use them to integrate and build new behaviours.

From this research, we aim to develop an Integrated Development Environment (IDE) that comprises a programming and compiling toolchain for swarm robots as a multi-agent system. The IDE is developed with three main characteristics that address the mentioned complexities in swarm programming. The ability of high-level algorithm composition is the main characteristic of the proposed IDE. It allows the users to program swarm behaviours in a graphical interface in a code-less approach. The swarm behaviours are designed based on a bottom-up design approach where the users can program more complex top-level behaviours using the low-level atomic behaviours. The behaviours are categorized into multiple levels, giving the users a clear understanding of combining and scaling them up without delving into low-level details. This abstraction can help manage the complexity of programming large swarms with sophisticated swarm behaviours with less effort.

The IDE automatically converts the graphical-level algorithm to a programming language which then facilitates the compilation process for creating binaries that can be executed on the robots supporting various hardware platforms. As the final characteristic, it supports uploading the binaries to the robots over the air (OTA). OTA programming refers to updating or reprograming hardware devices remotely without physical access. This capability is attained by wireless communication using WiFi, MQTT messaging protocol, and a central server which enables quick deployment of updates to multiple robots simultaneously and gives higher convenience, efficiency, and flexibility.

II. Related Work

We conducted the literature review based on two main categories: swarm programming tools-based studies and swarm behavioural algorithms-based studies. A summary of the studies discussed in the literature review, including the key concepts focused on, is shown in Figure 1.

A. Swarm Programming Tools and Frameworks

Research in Multi-Agent Systems (MAS) has recently led to the development of practical programming languages and tools appropriate for implementing such systems. Creating swarm intelligence programming frameworks has completely changed the field of swarm intelligence by enabling scientists to create and test sophisticated algorithms more quickly and effectively. Researchers can study and examine the behaviour of swarm algorithms using these frameworks' various features, which include visualization tools, simulation environments, and optimization algorithms. The following programming frameworks were well-identified in industrial usage and related robotics research.

Swarm-Bench [1] is a benchmarking framework designed to evaluate and compare swarm intelligence algorithms. It provides a standardized set of benchmark problems, performance metrics, and statistical analysis tools for assessing the effectiveness and efficiency of swarm-based optimization algorithms. Swarm-Bench facilitates fair comparisons between different algorithms by providing a common platform and metrics for evaluation. Since it primarily focuses on benchmarking and evaluation rather than algorithm implementation, it may require additional programming work to integrate custom algorithms.

SwarmOps [6] is a Python library for swarm intelligence optimization. It provides a set of algorithms inspired by swarm behaviour, such as particle swarm optimization (PSO) and differential evolution (DE). It focuses on providing a flexible and extensible framework for implementing and experimenting with swarm intelligence algorithms. SwarmOps offers various optimization techniques and allows users to customize algorithm parameters and problem domains. PySwarm [6] is a Python library focusing on particle swarm optimization (PSO) algorithms. It provides a simple and easy-to-use interface for implementing PSO and experimenting with different algorithm variants. PySwarm offers customization options for algorithm parameters, such as the number of particles, velocity update rules, and termination conditions. But this is less comprehensive compared to some other libraries. Both SwarmOps and PySwarm are limited to swarm intelligence optimization algorithms. SwarmLib [1] is a Java-based library for swarm intelligence algorithms. It offers a collection of swarm-based optimization techniques, including PSO, ant colony optimization (ACO), and artificial bee colony (ABC) algorithms. SwarmLib aims to provide comprehensive tools and algorithms for researchers and developers working on swarm intelligence optimization problems. But libraries like SwarmLib can be tested only with a pre-programmed set of swarm behaviours since those are just mere presentation tools rather than development libraries. And they don't discuss extending the simulation support to a real robot system which eventually is the end goal.

Physicomimetics [10], in the context of implementing swarm behaviours, involves utilizing a virtual physics framework to simulate and replicate the collective dynamics and behaviours observed in swarms of living organisms. It combines principles from physics and biology to create artificial systems that exhibit emergent properties and complex behaviours similar to those observed in natural swarms. It incorporates forces, velocities, accelerations, and collision detection concepts to

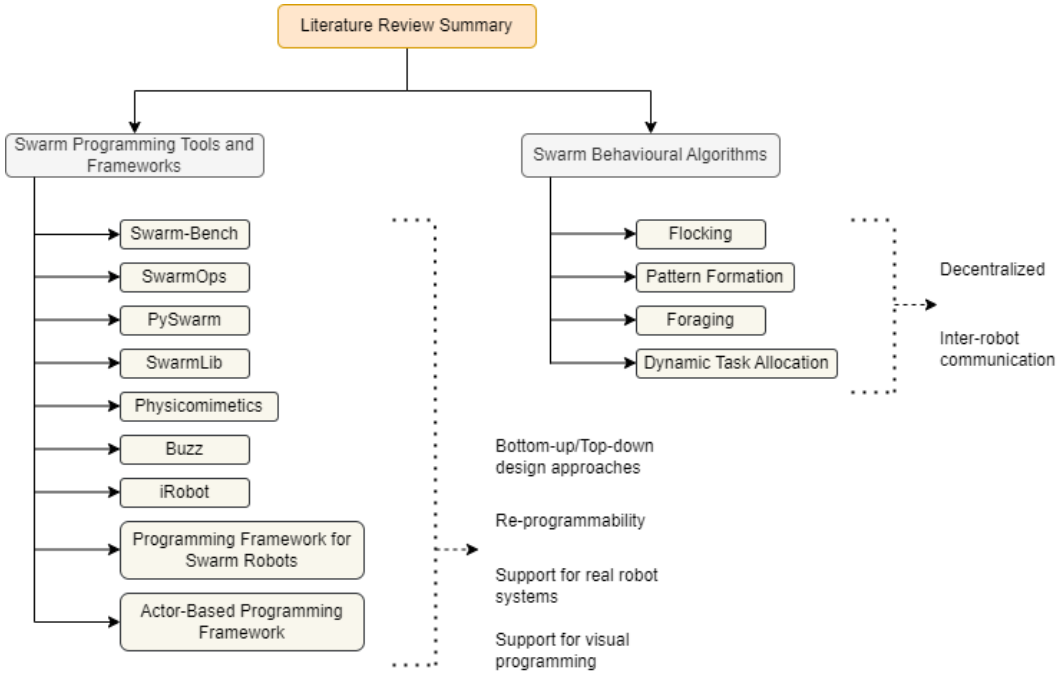

Fig. 1. Summary of the Literature Review.

simulate the physical constraints and interactions between the agents and their environment. This virtual physics framework typically operates based on predefined rules or algorithms that govern the behaviour of agents. And it is limited to a small set of behaviors, including pattern formation, pattern keeping, and obstacle avoidance. While this allows for the simulation of specific swarm behaviours, and they don't discuss the programmability of new behaviours, it may limit the system's adaptability and responsiveness to changes in the environment or swarm conditions. Real-world swarms exhibit adaptive and dynamic behaviours that may be challenging to capture solely within this framework.

Buzz [7] [1] is a programming language and simulation framework designed for developing and studying swarm robotics systems. It provides a high-level language for specifying swarm behaviours and a simulator for testing and analyzing swarm robotics algorithms. Buzz comes with three top-down programming primitives Swarm, Neighbours, and Virtual Stigmergy. It resolves the issue of hardware dependency using a virtual machine called BuzzVM. Buzz supports the coordination and communication among individual robots in a swarm, allowing the implementation of complex collective behaviours. But this is limited to coordination and communication aspects of swarm robotics.

iRobot [11] has been involved in research and development projects exploring swarm robotics, which focuses on coordinating and cooperating large groups of robots to achieve collective behaviours and tasks. One aspect of iRobot's approach to swarm robotics involves employing a bottom-up strategy. A bottom-up strategy in swarm robotics refers to a decentralized approach where individual robots, often called agents, interact with their local environment and each other based on simple rules or algorithms. The collective behaviour emerges from the interactions and coordination of these individual agents without the need for centralized control or explicit global instructions. This approach has applications in various domains, such as search and rescue, environmental monitoring, and distributed sensing, where a coordinated group of robots can accomplish tasks more efficiently and effectively than individual robots.

Similarly, another research done for implementing a programming framework for swarm robots [1] presented several built-in swarm robotic behaviours using the bottom-up design approach. It discusses the benefits of using the bottom-up rather than the top-down approach when designing swarm robotic behaviours. The framework offers the ability to integrate the built-in behaviours and create new ones. They were designed to be compatible with resource-limited hardware robotic platforms, which consisted only of a virtual pheromone-based communication system with IR sensors to measure the distance and the angles of incoming signals. The built-in behaviours were categorized into preliminary, pair, cluster, and global behaviours. The framework presented a wide range from preliminary behaviours such as moveForward, moveStop, and angularTurn to global-level behaviours such as aggregate and pattern formation. The tests were carried out on a custom-built simulation platform in Java for random movement with obstacle avoidance, object finding behaviour and aggregation behaviour.

The Actor-based programming framework [12], another research that follows a bottom-up design approach, offers several benefits. It allows task developers to model cooperative tasks

explicitly without getting caught up in the complexities of detailed robotic algorithms or brands. This framework reduces the workload on robotic algorithm developers by providing common functionalities. It has introduced the concept of an "Actor", representing high-level virtualization for robot platforms, enabling the management of collective behaviours, as each Actor maintains a data structure and is associated with different plug-in groups. A domain-specific language (DSL) is proposed for composing Actor-based tasks. This relieves task developers from the unnecessary details of individual robot manipulation, enabling them to focus on complex swarm robotic task coordination strategies. The framework is implemented in C++ and is validated through quantitative and qualitative methods, including simulations and in-field tests. While the Actor-based programming framework does not support visual programming for swarm robots, their objective of allowing the developers to focus on programming complex swarm behaviours without delving into low-level complexities aligns with our research goal.

While each tool or framework has its specific features and focuses, some common challenges or limitations that can be encountered in swarm intelligence-related tools. Some tools may focus on specific swarm intelligence algorithms, such as particle swarm optimization (PSO) or ant colony optimization (ACO) while neglecting other techniques. This limitation can restrict the range of problems addressed using the tool. As the complexity of the problem or the size of the swarm increases, the scalability and performance of the tools can become a challenge. The efficiency of the algorithms and the computational requirements may vary, impacting the practicality of using the tools for large-scale applications. While many tools provide default settings and parameters for the algorithms, the ability to easily customize and fine-tune the algorithms may vary. Some tools may have limited flexibility in adjusting algorithmic parameters or incorporating domain-specific knowledge. Integrating swarm intelligence tools with existing systems or frameworks can be challenging. Compatibility issues, dependencies, or lack of interoperability with other libraries or platforms may require additional effort or workarounds. Some tools, such as swarm robotics or blockchain-based decision-making, are designed with a specific domain or application in mind. While this specialization can be advantageous for specific use cases, it may limit the applicability of the tools in broader swarm intelligence contexts. Therefore, considering these common challenges is essential while evaluating and selecting a tool for your specific needs.

B. Swarm Behavioural Algorithms

When considering the studies related to swarm behavioural algorithms, one of such behaviours we encountered was the flocking behavioural algorithm. Flocks can be described as aggregations of many individuals which move together with cohesion, flexibility, and alignment. This phenomenon is observed in various species in nature, particularly among birds, fish, and certain insects.

In this area of research, Craig Reynolds [5] was amongst the first to abstract this flocking behaviour to a swarm of simulated birds which he called Boids. His implementation comprised three main behaviours: collision avoidance, flock centring, and velocity/heading matching. As a result, several approaches to adapt this behaviour to a robotic swarm have been made. Usually, these approaches needed stable communication channels between robots where heading data of its own is known, and headings or velocities of nearby robots or predefined leaders are communicated among robots which do not truly replicate the nature of a real flock.

Therefore, Christoph Moeslinger [4] and his colleagues came up with a low-end and easy-to-implement flocking algorithm which suits limited computational power and minimalist swarm robot equipment where it doesn't require communication, memory or global information. Their approach was discretising the robots' sensor fields into sectors and using different distance thresholds for attraction and repulsion in these sectors to achieve emergent alignment. The algorithm required four circumferential IR sensors and three discrete reactions in movement; move straight, turn left or turn right. The basic rules applied are collision avoidance, robot separation, flock cohesion, and emergent alignment. Tests were done for analysing flocking and aggregation capabilities using a simulator based on the multi-agent programming language NetLogo. The results showcased that both the mobility of a flock of swarm robots and aggregation time depended on the size of the flock, and the algorithm seemed to work well with comparatively small swarms.

Earlier approaches to pattern formation in swarm robotics have primarily focused on employing virtual forces and global knowledge to coordinate the robots and generate predefined patterns. However, these approaches often require complex algorithms and centralized control systems, which limit their adaptability and scalability. In contrast, the approach proposed by Mehmet Serdar Güzel and his team [8] introduces a new adaptive algorithm for pattern formation in swarm robotics. This algorithm allows the swarm to form a circle pattern autonomously, regardless of size, while providing collision prevention and adaptability in cluttered environments. The algorithm is integrated into a decentralized navigation system and has been successfully tested in a 3-D robotic simulator. The results of the experiments are encouraging, demonstrating the effectiveness of this algorithm and motivating further research in this direction.

The research paper [9] focuses on the foraging behaviour in swarm robotics and proposes different strategies for item searching. The study reviews the literature on foraging algorithms, mathematical models, and computer simulations. The researchers developed a state transition diagram, a logical flow chart, and algorithms for various item-searching strategies, including the expanding square, parallel sweep, and divider approaches. These strategies were simulated using the player/stage open-source simulation software. Statistical analysis was conducted to compare the strategies and identify the most effective ones. The results suggest that the parallel

sweep with divider policy yielded the best item detection time among the tested strategies.

The paper [15] introduces the Repulsion-Attraction (Rep-Att) algorithm for swarm robotics in the context of foraging tasks. Inspired by swarm intelligence observed in natural swarms, the algorithm utilizes direct communication among robots through sound signals. The algorithm incorporates repulsion signals to drive robots away from the nest in search of targets and attraction signals to invite other robots to exploit discovered target clusters. Simulation results demonstrate the effectiveness of the Rep-Att algorithm in improving foraging efficiency compared to random walk and repeller algorithms. Additionally, the paper presents experiments on implementing the communication model in hardware platforms, validating its functionality with realistic and noisy sound models. The algorithm's simplicity, low-cost hardware requirements, and utilization of bio-inspired navigation make it a promising approach for swarm foraging applications.

Dynamic task allocation, another important swarm behaviour, refers to the process of assigning tasks among individual robots in a swarm dynamically and adaptively. Wonki Lee and DaeEun Kim have proposed an algorithm [13] to handle dynamic task allocation in the object foraging task based on the decentralized strategy of the response threshold model without inter-robot communication. Each robot can select its task by regulating the response threshold value, which is updated based on the robot's local environment. The results have shown that the system can stochastically converge to the equilibrium of the desired task distribution and adapt the swarm to changes in the swarm members or task demands.

Another research study [14] focuses on the problem of task allocation in robotic systems. The paper explores different ways to solve this problem, mainly using evolutionary optimization algorithms. They suggest breaking down complex tasks into simpler ones and coordinating their execution as a solution. The paper highlights the importance of adapting task allocation to changes in the environment and swarm performance instead of using a centralized approach. They introduce various algorithms, classified based on their behavioural, market-based, and bio-inspired approaches. The algorithm proposed in the paper is a distributed task allocation algorithm based on Particle Swarm Optimization (PSO). It improves the allocation process by continuously updating the positions and velocities of the robots in the search space. The algorithm is designed to be resource-efficient and has been tested on a group of ELISA III robots. The paper presents experimental results and concludes that the proposed algorithm effectively achieves task allocation that meets the desired requirements. This demonstrates its effectiveness and efficiency in real-world situations.

REFERENCES

[1] Dassanayaka, M., Bandara, T., Adikari, N., Nawinne, I., & Ragel, R. (2020, July). A Programming Framework for Robot Swarms. In 2020 Moratuwa Engineering Research Conference (MERCon) (pp. 578-583). IEEE.

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

[3] Navarro, I., & Matía, F. (2013). An Introduction to Swarm Robotics. ISRN Robotics, 2013, 1–10.

[4] Moeslinger, C., Schmickl, T., & Crailsheim, K. (n.d.). LNAI 5778 - A Minimalist Flocking Algorithm for Swarm Robots.

[5] Reynolds, C.W. (1987). Flocks, herds, and schools. Computer Graphics, 21(4), 25–34.

[6] Madhushanka, H. M. K., & Perera, A. L. H. E. (2023). *Swarm Intelligence Programming Framework* Literature Review. February.

[7] Pinciroli, C., Lee-Brown, A., & Beltrame, G. (2015). Buzz: An Extensible Programming Language for Self-Organizing Heterogeneous Robot Swarms. http://arxiv.org/abs/1507.05946.

[8] Güzel, M.S., Gezer, E.C., Ajabshir, V.B., & Bostancı, E. (Year not specified). An Adaptive Pattern Formation Approach for Swarm Robots.

[9] Sakthivelmurugan, E., Senthilkumar, G., Prithiviraj, K.G., & Tinu Devraj, K.R. (Year not specified). Foraging behavior analysis of swarm robotics system.

[10] Spears, W.M., Spears, D.F., Heil, R., Kerr, W., & Hettiarachchi, S. (2005). An overview of physicomimetics. Lecture Notes in Computer Science, 3342, 84–97.

[11] McLurkin, J., Kaelbling, L.P. (1999). Stupid Robot Tricks: A Behavior-Based Distributed Algorithm Library for Programming Swarms of Robots.

[12] Yi, W., Di, B., Li, R., Dai, H., Yi, X., Wang, Y., & Yang, X. (Year not specified). An Actor-based Programming Framework for Swarm Robotic Systems.

[13] Lee, W., & Kim, D.E. (2019). Adaptive approach to regulate task distribution in swarm robotic systems. Swarm and Evolutionary Computation, 44, 1108–1118. https://doi.org/10.1016/j.swevo.2018.11.005

[14] Nedjah, N., De Macedo Mourelle, L. (2015). Pso-based distributed algorithm for dynamic task allocation in a robotic swarm

[15] Obute, Simon O., Dogar, Mehmet R., Dogar, Mehmet R. (2019): Simple Swarm Foraging Algorithm Based on Gradient Computation.