Cranfield UNIVERSITY

Inmo Jang

Effective Task Allocation Frameworks for Large-scale Multiple Agent Systems

School of Aerospace, Transport and Manufacturing,
Centre for Autonomous and Cyber-physical Systems

PhD

Cranfield UNIVERSITY

School of Aerospace, Transport and Manufacturing

Effective Task Allocation Frameworks for Large-scale Multiple Agent Systems

Inmo Jang

PhD Thesis

Academic Year 2015-2018

Supervisors: Dr Hyo-Sang Shin and Prof Antonios Tsourdos

October 2018

This thesis is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Abstract

This research aims to develop innovative and transformative decision-making frameworks that enable a large-scale multi-robot system, called *robotic swarm*, to autonomously address *multi-robot task allocation problem*: given a set of complicated tasks requiring cooperation, how to partition themselves into subgroups (or called *coalitions*) and assign the subgroups to each task while maximising the system performance. The frameworks should be executable based on local information in a decentralised manner, operable for a wide range of the system size (i.e., scalable), predictable in terms of collective behaviours, adaptable to dynamic environments, operable asynchronously, and preferably able to accommodate heterogeneous agents.

Firstly, for homogeneous robots, this thesis proposes two frameworks based on biological inspiration and game theories, respectively. The former, called *LICA-MC (Markov-Chan-based approach under Local Information Consistency Assumption)*, is inspired by fish in nature: despite insufficient awareness of the entire group, they are well-coordinated by sensing social distances from neighbours. Analogously, each agent in the framework relies only on local information and requires its local consistency over neighbouring agents to adaptively generate the stochastic policy. This feature offers various advantages such as less inter-agent communication, a shorter timescale for using new information, and the potential to accommodate asynchronous behaviours of agents. We prove that the agents can converge to a desired collective status without resorting to any global information, while maintaining scalability, flexibility, and long-term system efficiency. Numerical experiments show that the framework is robust in a realistic environment where information sharing over agents is partially and temporarily disconnected. Furthermore, we explicitly present the design requirements to have all these advantages, and implementation examples concerning travelling costs

minimisation, over-congestion avoidance, and quorum models, respectively.

The game-theoretical framework, called *GRAPE (GRoup Agent Partitioning and placing Event)*, regards each robot as a self-interested player attempting to join the most preferred coalition according to its individual preferences regarding the size of each coalition. We prove that selfish agents who have *social inhibition* can always converge to a *Nash stable partition* (i.e., a social agreement) within polynomial time under the proposed framework. The framework is executable based on local interactions with neighbour agents under a strongly-connected communication network and even in asynchronous environments. This study analyses an outcome's minimum-guaranteed suboptimality, and additionally shows that at least 50% is guaranteed if social utilities are non-decreasing functions with respect to the number of co-working agents. Numerical experiments confirm that the framework is scalable, fast adaptable against dynamical environments, and robust even in a realistic situation where some of the agents temporarily halt operation during a mission.

The two proposed frameworks are compared in the domain of *division of labour*. Empirical results show that LICA-MC provides excellent scalability with respect to the number of agents, whereas GRAPE has polynomial complexity but is more efficient in terms of convergence time (especially when accommodating a moderate number of robots) and total travelling costs. It also turns out that GRAPE is sensitive to traffic congestion, meanwhile LICA-MC suffers from slower robot speed. We discuss other implicit advantages of the frameworks such as mission suitability and additionally-built-in decision-making functions. Importantly, it is found that GRAPE has the potential to accommodate heterogeneous agents to some extent, which is not the case for LICA-MC.

Accordingly, this study attempts to extend GRAPE to incorporate the heterogeneity of agents. Particularly, we consider the case where each task has its minimum workload requirement to be fulfilled by multiple agents and the agents have different work capacities and costs depending on the tasks. The objective is to find an assignment that minimises the total cost of assigned agents while satisfying the requirements. GRAPE cannot be directly used because of the heterogeneity, so we adopt *tabu-learning heuristics* where an agent penalises its previously chosen coalition whenever it changes decision: this variant is called T-GRAPE. We prove that, by doing so, a Nash stable partition is always guaranteed to be determined in a decentralised manner. Experi-

mental results present the properties of the proposed approach regarding suboptimality and algorithmic complexity.

Finally, the thesis addresses a more complex decision-making problem involving team formation, team-to-task assignment, agent-to-working-position selection, fair resource allocation concerning tasks' minimum requirements for completion, and trajectory optimisation with collision avoidance. We propose an integrated framework that decouples the original problem into three subproblems (i.e., coalition formation, position allocation, and path planning) and deals with them sequentially by three respective modules. The coalition formation module based on T-GRAPE deals with a max-min problem, balancing the work resources of agents in proportion to the task's requirements. We show that, given reasonable assumptions, the position allocation subproblem can be solved efficiently in terms of computational complexity. For the path planning, we utilise an MPC-SCP (Model Predictive Control and Sequential Convex Programming) approach that enables the agents to produce collision-free trajectories. As a proof of concept, we implement the framework into a *cooperative stand-in jamming* mission scenario using multiple UAVs. Numerical experiments suggest that the framework could be computationally feasible, fault-tolerant, and near-optimal.

Comparison of the proposed frameworks for multi-robot task allocation is discussed in the last chapter regarding the desired features described at first (i.e., decentralisation, scalability, predictability, flexibility, asynchronisation, heterogeneity), along with future work and possible applications in other domains.

Acknowledgement

The research in this thesis would never have been as successful as it was without the support of many people. First, I would like to express my very great appreciation to my supervisors, Dr Hyo-Sang Shin and Prof Antonios Tsourdos, for giving me this wonderful opportunity to undertake this PhD research in Cranfield and for their great supervision, support, and encouragement on my research as well as on my personal life in the UK.

I also would like to thank my committee members, Prof Arthur Richards and Dr Adam Zagorecki, for reviewing my thesis and providing insightful suggestions to make this thesis much stronger. I am particularly grateful to Dr Namhoon Cho and Dr Chang-Hun Lee for having valuable discussions for my research whenever I needed them. Special thanks should be given to my friend Mr Sangjun Bae. With him, I had spent most of my time at the office and had enjoyed a variety of discussions, which were also very helpful for my research.

Finally, above all, I wish to thank my lovely wife Seonghyeon for being my greatest support and for tolerating many hardships we had encountered throughout this long journey. Also, I thank my dear son and daughter, Siyule and Sihah, for having been grown up very well in this new environment over the last three years.

Contents

List of Figures

List of Tables

Chapter 1

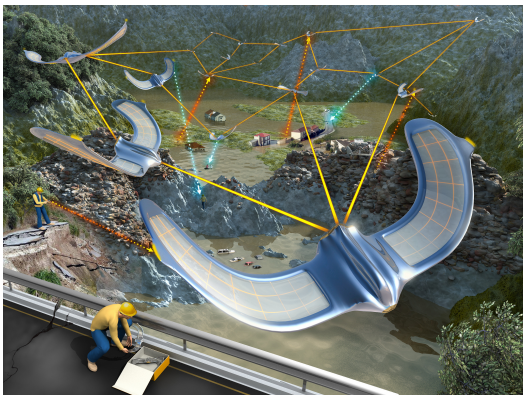
Introduction

1.1 Background and Motivation

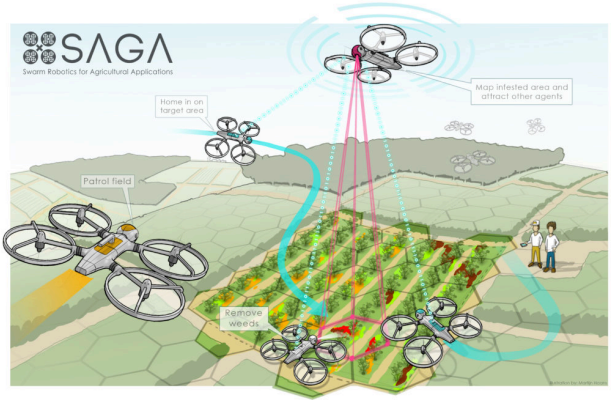
Cooperation of a large number of (possibly small-sized) robots, called *robotic swarm*, will play a major role in complex missions that existing operational concepts using a few of large robots could not deal with [1]. The individual robots (or called *agents*) are expected to be manufactured through mass production in lower cost with cheaper components, so each of them is likely to have limited capability to complete a single task alone [2]. Nevertheless, their cooperation will lead to successful outcomes because the system as a whole has promising advantages such as versatility, fault tolerance, flexibility, and improved performance through parallelism and redundancy [3–7]. Recently, Intel Corporation showed off an aerial light show of more than 200 quadcopters for CES 2018 in Las Vegas, USA [8], which has attracted considerable public attention. Other possible applications include environmental monitoring [9], ad-hoc network relay [10, 11], disaster management [12], object transportation [13], cooperative military missions [14], to name a few.

Due to the large cardinality of such a multi-agent system, however, it is infeasible for human operators to supervise each of them directly, but needed to entrust the swarm with certain levels of *autonomous decision-making* (e.g., task allocation, path planning, and individual control). Thereby, what only remains is to provide a high-level mission description, which is manageable by a few or even a single human operator. However,

(a)



(b)



(c)

Figure 1.1: Examples of the use of a swarm of aerial robots[1]: (a) Entertainment show [8] (b) Ad-hoc communication network [10]; (c) Agriculture applications [15]

there still exist various challenges in the autonomous decision-making of robotic swarms.

According to [3, 5, 7, 16, 17], a decision-making framework for large-scale multiple agent systems (or called *swarm intelligence framework*) should be

[1]Images downloaded in February 2018 from (a) `http://www.bbc.co.uk/news/av/embed/p05th5q4/42643790`, (b) `https://flic.kr/p/BFuhxR`, and (c) `http://echord.eu/saga/`, respectively.

- Decentralised : A desired collective behaviour can be achieved not by any central control unit but by individual autonomous agents who make decisions based on local information or local interactions with neighbour agents.

- Scalable : The framework is operable for a wide range of the system size (e.g, the number of agents).

- Predictable : Human operators can estimate the quality of a collective outcome given by the framework, for example, its suboptimality in terms of certain performance indices and convergence time towards a desired global behaviour.

- Flexible (or adaptable) : The framework can quickly adapt to any dynamic changes in the environment (e.g., unexpected elimination or addition of some agents or tasks).

- Operable in asynchronous environments: Due to the large cardinality of the system and its decentralisation, it is challenging for all the given agents to execute decision-making procedures synchronously. For synchronisation in practice, "artificial delays and extra communication must be built into the framework" [17], which may cause considerable inefficiency on the system. Hence, the framework should be operable even in asynchronous environments.

- (Optionally) Able to accommodate different interests of agents: Although a robotic swarm typically consists of homogeneous agents [18], in some cases, individual agents may have different levels of interest. For example, some of them "may place a higher utility on successful completion of tasks, while others are obligated to participate, but wish to conserve resource" [19]. This situation probably happens when the agents are "designed, owned, or operated by several individuals or organisations that may have different goals" [20, 21] or even when the agents are physically identical but have different energy resources. In a sense, such agents can be regarded as heterogeneous, and accommodating their different interests would be desirable to yield higher system performance.

The lack of such decision-making frameworks is one of the major factors hindering robotic swarm implementation [22, 23].

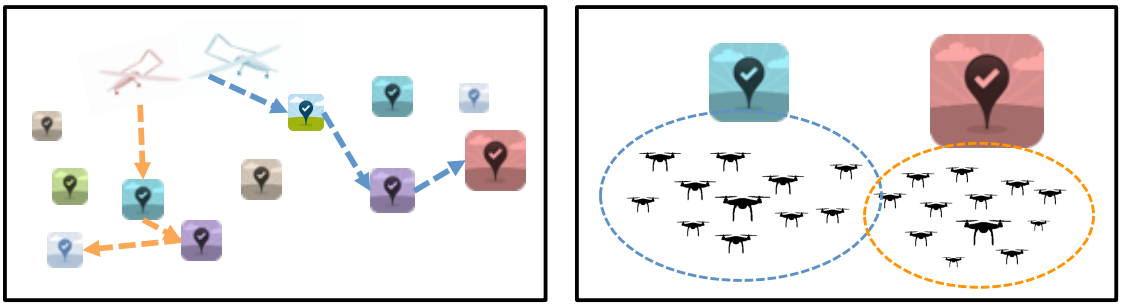


Figure 1.2: Two types of multi-robot task allocation problems

1.2 Research Aim and Objectives

Research Aim

This research attempts to develop innovative and transformative decision-making frameworks that have all the desired features mentioned above (i.e., decentralised, scalable, predictable, adaptable, asynchronous, and possibly able to accommodate agents' different interests) to operate a large-scale multi-agent system effectively. Amongst autonomous decision-making problems, this study primarily addresses *multi-robot task allocation problems* (MRTA). Depending on the characteristics of given agents and tasks, MRTA can be categorised into various types [24, 25], and Figure 1.2 briefly illustrates main two types amongst them. The left-side case, which has been addressed relatively more over the last decade, is characterised by the following research question: given a few of individually highly capable robots and a relatively larger number of tasks, how to make a bundle of tasks for a each robot. On the contrary, when it comes to a *robotic swarm*, a question arises differently as shown in the right subfigure: how to partition a set of agents into subgroups and assign the subgroups to each task, while maximising the system performance. To address this particular type of MRTA (or referred to as *coalition formation problem* [26]) while obtaining all the desirable features as much as possible is the main aim of this thesis.

Objectives

As promising methodologies, game-theoretical and bio-inspired approaches are to be investigated because these methods innately consider autonomous agents as well as have a great potential in terms of adaptability and scalability. Specific objectives of this study are presented as follows:

1. **Development of a bio-inspired framework:** As the first task of this research, existing literature on task allocation problems for robotic swarms is to be reviewed to identify a research gap that should have been considered for realistic environments. This gap is to be addressed by a novel framework inspired from the biological swarms such as the flocks of birds, the colonies of bees, or the school of fish. It is well known that these living creatures have insufficient awareness of the entire group but show well-harmonised collective behaviours in nature. Analogously, in our proposed framework, each agent should behave based on local information and local interactions with its neighbours with very simple decision-rules, while a desired collective behaviour emerges.

2. **Development of a game-theoretical framework:** A novel framework is to be designed through the introduction of game theories that can model conflicts of multiple robots in a task allocation problem. Standing on the shoulder of the giant (i.e., game theories), which is "the study of mathematical models of conflict and cooperation between intelligent rational decision-makers" [27], this study will analyse collective behavioural characteristics of a robotic swarm operated under the proposed framework. For example, suboptimality of an outcome given by the framework is to be investigated as analogous to that of a *Nash equilibrium* (i.e., a stable social outcome in game theories), called the *price of anarchy*.

3. **Comparison between the proposed methodologies:** The two proposed frameworks are to be compared with each other, after a fair mission scenario is formalised, regarding scalability, versatility, and other inherent pros and cons.

4. **Extension to accommodate heterogeneous agents:** One of the proposed frameworks is to be extended so that it can incorporate the heterogeneity of

agents. Even if a swarm of robots were identically manufactured through mass production, each agent might possess a different level of energy resource in practice during a mission, which gives rise to heterogeneity. Furthermore, we will also consider heterogeneous tasks and their minimum requirements for completion.

5. **Extension to address additional decision-making issues:** When implementing the proposed frameworks into a cooperative mission, we must encounter further practical issues in decision-making on top of the task allocation problem shown in Figure 1.2. Amongst them, this research particularly will address position allocation (i.e., exactly which position to go for executing the assigned task) and path planning (i.e., how to reach the assigned positions without collisions) problems (please refer to Figure 1.5), and propose an integrated decision-making framework for addressing all these problems.

6. **Implementation of the proposed methodologies into practical scenarios:** The proposed frameworks are to be implemented to realistic mission scenarios that require autonomous decision-making of a large-scale multi-agent system.

In this thesis, for each proposed framework, the desired features introduced in Section 1.1 are to be mostly formally analysed with support from numerical experiments. The inter-agent communication network assumed here is *strongly-connected* (i.e., there exists a directed communication path between any two arbitrary agents), but it may not be the case in reality. Thus, we will numerically validate the performances of the proposed frameworks in the case where some of the given agents can not operate or not communicate with others temporarily.

1.3 Contribution to Knowledge

Contributions of the thesis, found by addressing the prescribed objectives, are as follows. Each contribution is illustrated as a block in Figure 1.3, which also shows how all the blocks are related to each other in the thesis. Please note that every block was submitted to or published in a journal or peer-reviewed conference proceedings, and their detailed list will be shown in Section 1.5.

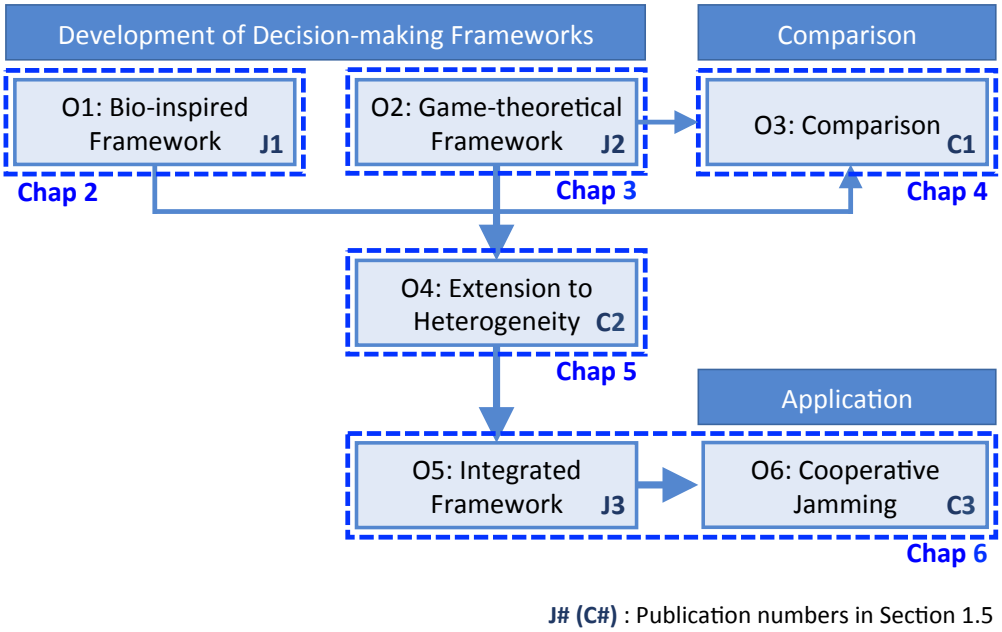


Figure 1.3: The outline of the thesis

1.3.1 A Bio-inspired Framework

For a swarm of homogeneous robots, the multi-robot task allocation problem considered can be reduced to a problem of how to distribute robotic labours over given tasks according to the desired labour distribution. This problem is called *swarm distribution guidance problem*, which can be briefly illustrated in Figure 1.4. This study proposes a Markov-chain-based framework in which population fractions of a swarm are modelled as the system state, and each agent behaves stochastically according to a time-inhomogeneous Markov matrix depending on the difference from the current swarm status to the desired status. Unlike most of the existing frameworks handling this problem, the proposed framework suggests utilising *local* information available from neighbours to adjust the stochastic policies, inspired by a swarm of fish. Accordingly, as each agent requires only local consistency on information with neighbouring agents, not the global consistency, the proposed framework offers various advantages, e.g., less inter-agent communication, a shorter timescale for using new information, and the potential to incorporate an asynchronous decision-making process. We prove that, even using

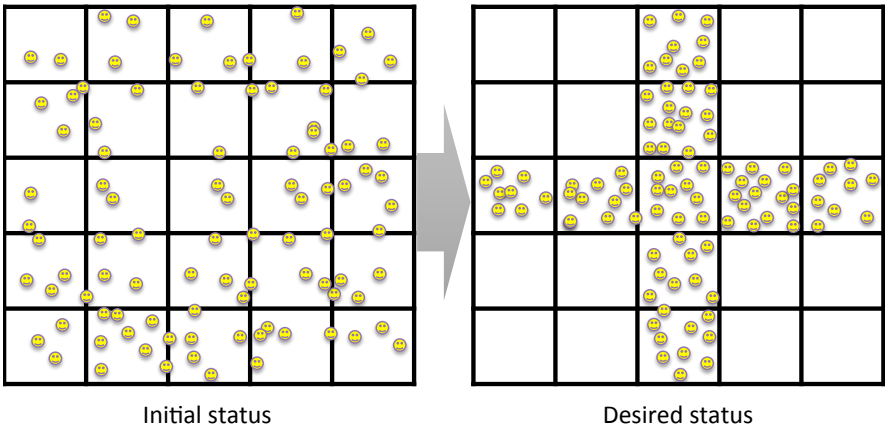


Figure 1.4: Swarm Distribution Guidance Problem: How to distribute homogeneous agents into tasks (or bins), satisfying a desired population fraction for each task

such insufficient information, the framework guarantees convergence towards the desired labour distribution while maintaining advantages of existing global-information-based approaches. The design requirements to hold the convergence and those advantages are explicitly found. This study also presents implementation examples concerning travelling cost minimisation, over-congestion avoidance, and quorum models, respectively. Numerical experiments confirm the effectiveness of the proposed framework and particularly show its improved robustness in a partially communication-disconnected situation, compared with a recent existing work based on global information [16]. The detail is included in Chapter 2.

1.3.2 A Game-theoretical Framework

This study allows agents to be self-interested so that they have tendency to make coalitions according to individual preferences regarding the size of each coalition. This situation undoubtedly induces conflicts between them, but we desire cooperation of such selfish agents. To this end, this study proposes a novel game-theoretical decentralised coordination framework based on *anonymous hedonic games* [28]. Interestingly, we prove that selfish agents who have social inhibition can always converge to a *Nash stable partition* (i.e., social agreement) within polynomial time $O(n_a^2 d_G)$, where n_a is the

number of agents and d_G is the graph diameter of the agents' communication network. The proposed framework is straightforward and executable based on local interactions with neighbour agents under a strongly-connected communication network and even in asynchronous environments. This study analytically presents a mathematical formulation for computing the lower bound of a converged outcome's suboptimality, and additionally shows that 50% of suboptimality can be at least guaranteed if social utilities are non-decreasing functions with respect to the number of co-working agents. Through numerical experiments, it is confirmed that the proposed framework is scalable, fast adaptable against dynamical environments, and robust even in a situation where some random agents temporarily somehow do not operate during a mission. The detail is presented in Chapter 3.

1.3.3 A Comparative Study of the Two Proposed Frameworks

This study compares the two swarm intelligence frameworks previously introduced: the Markov-Chain-based approach under *Local Information Consistency Assumption*, called *LICA-MC*, and the game-theoretical approach, called *GRAPE*. For this comparison, we implement both frameworks into *swarm distribution guidance problem*, shown in Figure 1.4, and then perform numerical experiments with various environmental settings. The statistical results show that LICA-MC provides excellent scalability regardless of the number of robots, whereas GRAPE is more efficient regarding convergence time (especially when accommodating a relatively fewer number of swarm robots) and total travelling costs. Furthermore, this study investigates other implicit advantages of the frameworks such as mission suitability, additionally-built-in decision-making functions, and sensitivity to traffic congestion or robots' mobility. Importantly, it is found that GRAPE has the potential to accommodate heterogeneous agents to some extent, which is not the case for LICA-MC. The detail is included in Chapter 4.

1.3.4 Consideration of Heterogeneous Agents

This study attempts to extend GRAPE to accommodate heterogeneous agents because of its potential found in Chapter 4. Particularly, we consider the case where each task has a different type of minimum workload requirement to be fulfilled by multiple agents, and the agents have different work capacities and costs depending on the tasks. The objective is to find an assignment that minimises the total cost of assigned agents while satisfying the requirements of the tasks: this optimisation problem would be attractive in a business application because it is desirable to reduce unnecessary costs but comply with customers' requirements. We formulate this problem as *the minimisation version of the generalised assignment problem with minimum requirements* (MinGAP-MR). However, due to the heterogeneity, it is not possible to directly use GRAPE for the problem considered. Thus, we suggest adopting tabu-learning heuristics where an agent penalises its previously chosen coalition whenever it changes a decision (this variant is called *T-GRAPE*), and show that this approach guarantees convergence of heterogeneous agents towards a Nash stable partition. Numerical experiment results present the performances of the proposed approach in terms of suboptimality and algorithmic complexity. The detail is shown in Chapter 5.

1.3.5 An Integrated Framework and Its Application to Cooperative Jamming of UAVs

For a cooperative mission consisting of multiple spatially-distributed tasks, a robotic swarm's practical decision-making problem includes team formation, team-to-task assignment, agent-to-work-position assignment, and trajectory optimisation with collision avoidance. The problem becomes even more complicated when involving heterogeneous agents, tasks' minimum requirements, and fair allocation. This study proposes an integrated approach that approximates the complex original problem into three subproblems (i.e., coalition formation, position allocation, and path planning) and addresses them sequentially by three different proposed modules. The coalition formation module based on T-GRAPE deals with a max-min problem, the objective of which is to

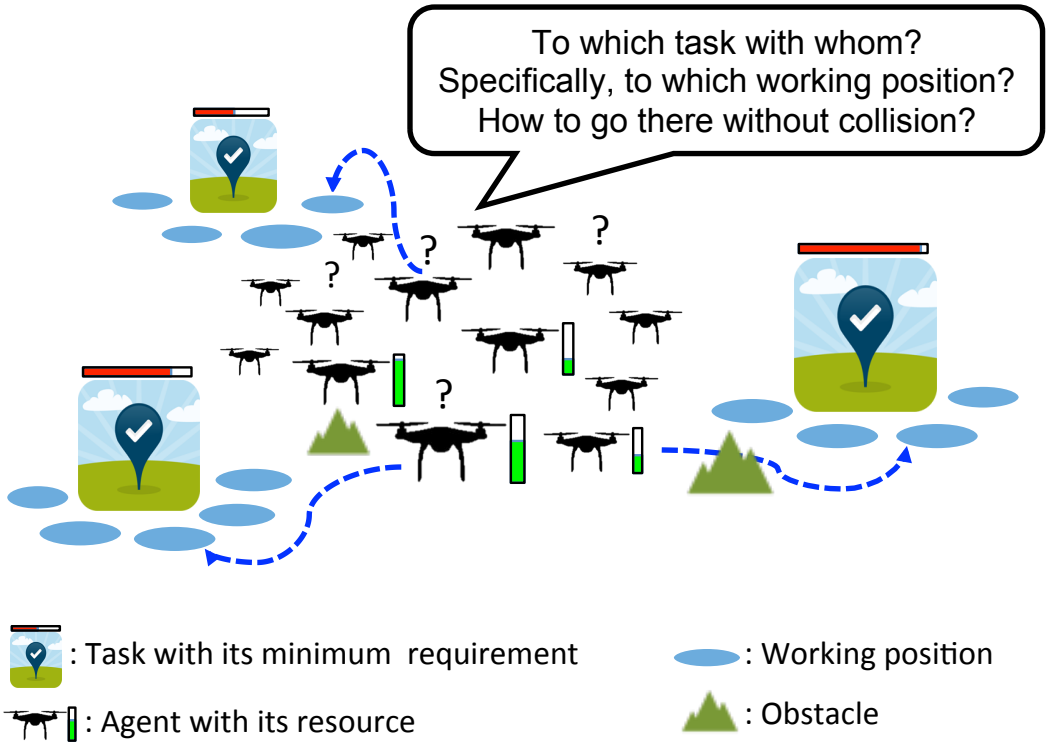


Figure 1.5: A brief illustration of the decision-making issues considered in Chapter 6

partition the agents into disjoint task-specific teams in a way that balances the agents' work resources in proportion to the task's minimum workload requirements. For agents assigned to the same task, given reasonable assumptions, the position allocation sub-problem can be efficiently addressed in terms of computational complexity. For the trajectory optimisation, an MPC-SCP (Model Predictive Control and Sequential Convex Programming) algorithm is utilised, which reduces the size of the problem so that the agents can generate collision-free trajectories on a real-time basis. As a proof of concept, we implement the framework into a cooperative stand-in jamming mission scenario and show its feasibility, fault tolerance, and near-optimality based on numerical experiment. The detail is included in Chapter 6.

1.4 Organisation of the Thesis

This thesis is organised as follows (also as shown in Figure 1.3). Chapter 2 and Chapter 3 propose the bio-inspired framework based on Markov process and the game-theoretical framework based on anonymous hedonic games, respectively. Chapter 4 compares the two methodologies in terms of labour division of a robotic swarm and discuss their pros and cons. Chapter 5 extends the game-theoretical framework to accommodate heterogeneous agents. Based on this work, Chapter 6 presents the integrated framework that also addresses position allocation and path planning issues, and shows its implementability in a cooperative jamming mission using multiple UAVs. Lastly, the thesis ends with conclusions and provides suggestions for future work in Chapter 7. Please note that the thesis is *paper-format*, consisting of individual papers submitted as its chapters.

1.5 The List of Published/Submitted Works

The following papers were submitted or published in relation to this PhD research. Note that although conference paper C4 is not first-authored by myself, I contributed to it in algorithm development and problem formulation.

Journal Papers

J1. **I. Jang**, H.S. Shin, A. Tsourdos, "Local Information-Based Control for Probabilistic Swarm Distribution Guidance," *Swarm Intelligence* (resubmitted after minor revision)

J2. **I. Jang**, H.S. Shin, A. Tsourdos, "Anonymous Hedonic Game for Task Allocation in a Large-Scale Multiple Agent System," *IEEE Transactions on Robotics* (in press)

J3. **I. Jang**, H.S. Shin, A. Tsourdos, J.Jeong, S. Kim, J.Suk, "An Integrated Decision-making Framework of a Heterogeneous Aerial Robotic Swarm for Cooperative

Tasks with Minimum Requirements," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* (in press)

Peer-reviewed Conference Papers

C1. **I. Jang**, H.S. Shin, A. Tsourdos, "A Comparative Study of Game-theoretical and Markov-chain-based Approaches to Division of Labour in a Robotic Swarm," *IFAC Aerospace Controls TC Workshop Networked & Autonomous Air & Space Systems*, Santa Fe, NM, USA, 13–15 Jun 2018

C2. **I. Jang**, H.S. Shin, A. Tsourdos, "A Game-theoretical Approach to Heterogeneous Multi-Robot Task Assignment Problem with Minimum Workload Requirements," *The 4th Workshop on Research, Education and Development of Unmanned Aerial Systems*, Linköping, Sweden, 3–5 October 2017

C3. **I. Jang**, J. Jeong, H.S. Shin, S. Kim, A. Tsourdos, J. Suk, "Cooperative Control for a Flight Array of UAVs and an Application in Radar Jamming," *The 20th World Congress of the International Federation of Automatic Control*, Toulouse, France, 9–14 July 2017

C4. H.S. Shin, **I. Jang**, A. Tsourdos, "Frequency Channel Assignment for Networked UAVs using a Hedonic Game," *The 4th Workshop on Research, Education and Development of Unmanned Aerial Systems*, Linköping, Sweden, 3–5 October 2017

References

[1] H.-S. Shin and P. Segui-Gasco, "UAV Swarms: Decision-Making Paradigms," in *Encyclopedia of Aerospace Engineering*. John Wiley & Sons, 2014, pp. 1–13.

[2] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable Self-assembly in a Thousand-robot Swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[3] E. Sahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 10–20.

[4] I. Navarro and F. Matía, "An Introduction to Swarm Robotics," *ISRN Robotics*, vol. 2013, pp. 1–10, 2013.

[5] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm Robotics: a Review from the Swarm Engineering Perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[6] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot Task Allocation: A Review of the State-of-the-Art," in *Cooperative Robots and Sensor Networks 2015*. Cham: Springer International Publishing, 2015, vol. 604, pp. 31–51.

[7] M. Dorigo, M. Birattari, and M. Brambilla, "Swarm robotics," *Scholarpedia*, vol. 9, no. 1, p. 1463, 2014.

[8] BBC News, "CES 2018: Intel's swarm of drones lights up Vegas night sky," 2018. [Online]. Available: http://www.bbc.co.uk/news/av/technology-42643790/ces-2018-intel-s-swarm-of-drones-lights-up-vegas-night-sky

[9] K. Barton and D. Kingston, "Systematic Surveillance for UAVs: A Feedforward Iterative Learning Control Approach," in *American Control Conference*, Washington, DC, USA, 2013, pp. 5917–5922.

[10] S. Hauert, J.-C. Zufferey, and D. Floreano, "Evolved swarming without positioning information: an application in aerial communication relay," *Autonomous Robots*, vol. 26, no. 1, pp. 21–32, 2009.

[11] I. Bekmezci, O. K. Sahingoz, and S. Temel, "Flying Ad-Hoc Networks (FANETs): A Survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[12] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz, "Help from the Sky: Leveraging UAVs for Disaster Management," *IEEE Pervasive Computing*, vol. 16, no. 1, pp. 24–32, 2017.

[13] Jianing Chen, M. Gauci, Wei Li, A. Kolling, and R. Gros, "Occlusion-Based Cooperative Transport with a Swarm of Miniature Mobile Robots," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 307–321, 2015.

[14] M. Mayer, "The new killer drones: understanding the strategic implications of next-generation unmanned combat aerial vehicles," *International Affairs*, vol. 91, no. 4, pp. 765–780, 2015.

[15] D. Albani, D. Nardi, and V. Trianni, "Field coverage and weed mapping by UAV swarms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vancouver, BC, Canada, 2017, pp. 4319–4325.

[16] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1103–1123, 2017.

[17] L. Johnson, S. Ponda, H.-L. Choi, and J. How, "Asynchronous Decentralized Task Allocation for Dynamic Environments," in *Infotech@Aerospace 2011*, St.Louis, MO, USA, 2011.

[18] B. Khaldi and F. Cherif, "An Overview of Swarm Robotics : Swarm Intelligence Applied to Multi-robotics," *International Journal of Computer Applications*, vol. 126, no. 2, pp. 31–37, 2015.

[19] C. E. Pippin and H. Christensen, "Cooperation based dynamic team formation in multi-agent auctions," in *Proc. of SPIE*, vol. 8389, 2012, pp. 838 919–838 919–8.

[20] C. M. Clark, R. Morton, and G. A. Bekey, "Altruistic relationships for optimizing task fulfillment in robot communities," *Distributed Autonomous Robotic Systems 8*, pp. 261–270, 2009.

[21] R. D. Morton, G. A. Bekey, and C. M. Clark, "Altruistic task allocation despite unbalanced relationships within Multi-Robot Communities," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, St.Louis, MO, USA, oct 2009, pp. 5849–5854.

[22] Robotics Trends, "4 Challenges Holding Back Swarm Robotics," 2015. [Online]. Available: http://www.roboticstrends.com/article/4_challenges_holding_back_swarm_robotics

[23] M. Chamanbaz, D. Mateo, B. M. Zoss, G. Tokic, E. Wilhelm, R. Bouffanais, and D. K. P. Yue, "Swarm-Enabling Technology for Multi-Robot Systems," *Frontiers in Robotics and AI*, vol. 4, pp. 1–18, 2017.

[24] B. P. Gerkey and M. J. Matarić, "A Formal Analysis and Taxonomy of Task Allocation in Multi-robot Systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[25] G. A. Korsah, A. Stentz, and M. B. Dias, "A Comprehensive Taxonomy for Multi-robot Task Allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[26] O. Shehory and S. Kraus, "Methods for Task Allocation via Agent Coalition Formation," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 165–200, 1998.

[27] R. B. Myerson, *Game Theory: Analysis of Conflict*. Cambridge, Massachusetts, USA: Harvard Univercity Press, 1997.

[28] A. Darmann, "Group Activity Selection from Ordinal Preferences," in *Algorithmic Decision Theory. ADT 2015. Lecture Notes in Computer Science*, ser. Lecture Notes in Computer Science, T. Walsh, Ed. Berlin, Heidelberg: Springer Cham, 2015, vol. 9346, pp. 35–51.

Chapter 2

Bio-Inspired Local Information-Based Control for Probabilistic Swarm Distribution Guidance

2.1 Introduction

This chapter addresses a task allocation problem for a large-scale multiple-robot system, called *robotic swarm*. In the problem considered, individual agents are assumed to be homogeneous since a swarm is usually realised through mass production [1]. In this context, the task allocation problem becomes a *swarm distribution guidance problem* [2–4], which can be illustrated as in Figure 2.1. As shown in the figure, the swarm distribution problem is about how to distribute a swarm of agents into given tasks, also called bins, to achieve the desired population fraction (or swarm density) for each task.

For swarm distribution guidance problems, there have been two main approaches widely studied: probabilistic approaches based on Markov chains [2–10] or differential equations [7–13] have been widely studied. These approaches generally focus not on individual agents, but on their ensemble dynamics. It is the reason why they are often called *Eulerian* [9, 14] or *macroscopic* frameworks [15, 16]. In these approaches, swarm densities over given bins are represented as system states. A state-transition matrix for the states describes *stochastic (decision) policies*, i.e., the probabilities that agents in a

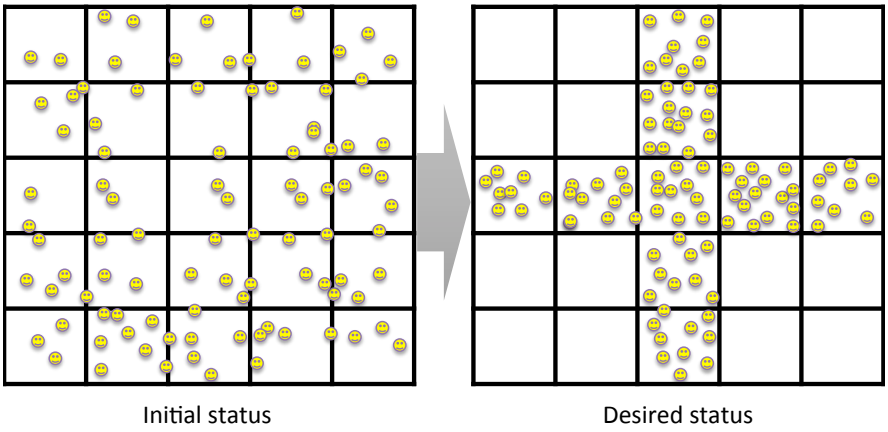


Figure 2.1: Swarm Distribution Guidance Problem: How to distribute homogeneous agents into tasks (or bins), satisfying a desired population fraction for each task

bin switch to another within a time unit. Accordingly, individual agents make decisions according to the policies in a random, independent and memoryless manner.

These approaches can be classified into two framework groups: *open-loop-type* frameworks [2–5,16–18] and *closed-loop-type* frameworks [7–13]. Agents under open-loop-type frameworks are controlled by time-invariant stochastic policies. The policies, which make a swarm converge to a desired distribution, are pre-determined by a central controller and broadcasted to each agent before the mission begins. Communication between agents is hardly required during the mission, and thus the communication complexity is minimised. However, the agents only have to follow the predetermined policies without incorporating any feedback, and thus there still remain some agents who unnecessarily and continuously move around bins even after the swarm reaches the desired status. Therefore, the trade-off between convergence rate and long-term system efficiency becomes critical in these frameworks [18].

Closed-loop-type frameworks allow agents to adaptively construct their own stochastic policies at the expense of communicating with other agents to perceive the concurrent swarm status. Based on such information, the agents can synthesise a time-inhomogeneous transition matrix to achieve certain objectives and requirements: for example, maximising convergence rates [8], minimising travelling costs [10], and temporarily adjusting the policies when bins are overpopulated or underpopulated than certain levels [11, 12]. In particular, Bandyopadhyay et al. [10] recently proposed a

closed-loop-type algorithm that exhibits faster convergence as well as less undesirable transition behaviours, compared with an open-loop-type algorithm. This algorithm can mitigate the issue with trade-off that is critical in open-loop-type frameworks.

Under these backgrounds, this chapter aims to develop a closed-loop-type framework for the swarm distribution guidance. To the best of our knowledge, most of the existing closed-loop-type algorithms are based on *Global Information Consistency Assumption* (GICA) [19]. GICA implies that the information necessary to generate time-varying stochastic policies is about the global swarm status (i.e., *global* information), and it also needs to be consistently known by all agents. Achieving GICA requires each agent to somehow interact with all the others through a multi-hop fashion and it "happens on a global communication timescale" [19].

The main focus of this work is to relax GICA: we propose a close-loop-type framework that relied on *Local Information Consistency Assumption* (LICA), i.e., *local* information only needs to be consistently known by the *local agent groups*. Note that the proposed LICA-based framework utilises *local* information as its feedback gains. In fact, the existing closed-loop-type methods in [10, 13] do not necessarily require every agent to perceive global knowledge exactly, while providing a graceful performance degradation even if local estimates of the global information are used. However, the key difference from the previous works is that the proposed LICA-based framework utilises *local information* as its feedback gains. In this chapter, we will use the term "local information" to refer to the knowledge available to all agents in the same bin and those obtainable via communication with other agents in all *neighbour bins* (defined in Definition 7). This is inspired by decision-making mechanisms of a fish swarm, where each fish adjusts its individual behaviour based on those of its neighbours [20–23]. Analogously, each agent in the proposed framework uses its local status to generate its stochastic policies.

The proposed framework based on LICA facilitates various advantages that could not be achievable in a GICA-based framework, while retaining the aforementioned advantages in the GICA-based framework. The first obvious benefit of the LICA-based framework is reduction in inter-agent communication required for decentralised decision-making, as can be seen from the experimental results in Figure 2.6(d). The proposed approach consequently "provides a much shorter timescale for using new infor-

mation because agents are not required to ensure that this information has propagated to the entire team before using it" [19]. Moreover, exploiting LICA in the close-loop-type framework enables asynchronous implementation of the framework (as shown in Section 2.5) and exhibits the robustness against dynamical changes in bins and also in agents (Section 2.6.3). Furthermore, agents in the proposed method do not need to know global mission knowledge such as a desired distribution *a priori* (i.e., before a mission starts), which is the case in the recent works [10, 13], as long as they can sense all the neighbour bins' desired swarm densities in an impromptu manner.

The stability and performance of the proposed framework are investigated via theoretical analysis and empirical tests. In the theoretical analysis, we prove that agents in the framework developed asymptotically converge towards the desired swarm distribution, even using local-information-based feedback. Also, the theoretical analysis provides the design requirements for the time-inhomogeneous Markov chain to achieve all the benefits discussed. Empirical tests demonstrate the performance of the proposed framework in three implementation examples: 1) travelling cost minimisation; 2) convergence rate maximisation under flux upper limits; and 3) quorum-based policies generation (similar to [11, 12]). Moreover, we present an asynchronous version of the proposed framework, which provides more robustness against temporary network disconnection of partial agents, compared with the recent work [10].

The rest of the chapter is organised as follows. Section 2.2 introduces essential definitions and notations of a Markov-chain-based approach. Section 2.3 describes the desired features for swarm distribution guidance, proposes a closed-loop-type framework with its design requirements, and performs theoretical analysis. We provide examples of how to exploit the framework for specific problems in Section 2.4 and asynchronous implementation in Section 2.5. Numerical experimental results are provided in Section 2.6, followed by concluding remarks in Section 2.7.

Notations

$\emptyset$, $\mathbf{0}$, I and $\mathbf{1}$ denote the empty set, the zero matrix of appropriate sizes, the identity matrix of appropriate sizes, and a row vector with all elements are equal to one, respectively. $\mathbf{v} \in \mathbb{P}^n$ is a $1 \times n$ row-stochastic vector such that $\mathbf{v} \geq \mathbf{0}$ and $\mathbf{v} \cdot \mathbf{1}^\top = 1$. $\mathbf{v}[i]$

Table 2.1: Nomenclature

Symbol	Description
$\mathcal{B}_j$	The j-th bin amongst a set of n_b bins (Definition 1)
$\mathcal{A}$	A set of n_a agents (Definition 1)
$\mathbf{x}_k$	The current (global) swarm distribution (Definition 4)
M_k	Stochastic policy of the agents (Definition 5)
Θ	The desired swarm distribution (Definition 6)
$\mathbf{A}_k$	Physical motion constraint matrix (Definition 2)
$\mathcal{N}_k(j)$	A set of neighbour bins of the j-th bin (Definition 7)
$\mathcal{A}_{\mathcal{N}_k(j)}$	A set of agents in $\mathcal{N}_k(j)$
$\mathbf{n}_k[j]$	The number of agents in $\mathcal{B}_j$ at time instant k (Eqn. (2.3))
$\bar{\mathbf{x}}_k[j]$	The current *local* swarm density at the j-th bin (Eqn. (2.3))
$\bar{\Theta}[j]$	The locally-desired swarm density at the j-th bin (Eqn. (2.4))
P_k	Primary guidance matrix (Eqn. (2.9))
S_k	Secondary guidance matrix (Eqn. (2.9))
$\bar{\xi}_k[j]$	Primary local-feedback gain (e.g., Eqn. (2.5))
$G_k[j]$	Secondary local-feedback gain (Eqn. (2.8))

indicates the i-th element of vector $\mathbf{v}$. Prob(E) denotes the probability that event E will happen. $\odot$ denotes the Hadamard product.

2.2 Preliminaries

This section provides the basic concept of a Markov-chain-based approach and presents definitions and assumptions necessary for our proposed framework, which will be shown in Section 2.3. Note that most of them are embraced from the existing literature [8,10].

Definition 1 (*Agents and Bins*)**.** A set of n_a *homogeneous agents* $\mathcal{A} = \{a_1, a_2, ..., a_{n_a}\}$ are to be distributed over a prescribed region in a state space $\mathcal{B}$. The entire space is partitioned into n_b disjoint *bins* such that $\mathcal{B} = \cup_{j=1}^{n_b} \mathcal{B}_j$ and $\mathcal{B}_j \cap \mathcal{B}_l = \emptyset$, $\forall l \neq j$. We also regard $\mathcal{B} = \{\mathcal{B}_1, ..., \mathcal{B}_{n_b}\}$ as the set of all the bins. Each bin $\mathcal{B}_j$ represents a predefined range of an agent's state, e.g., position, task assigned, behaviour, etc. Note that binning of the state space should be done problem-specifically in a way that accommodates all the following assumptions and definitions in this section. For example, the bin sizes are not necessarily required to be uniform, but can vary depending on physical constraints of the space or communication radii of given agents.

Definition 2 (*Agent motion constraint*)**.** The agent motion constraints over the given bins $\mathcal{B}$ are represented by $\mathbf{A}_k \in \{0,1\}^{n_b \times n_b}$, where $\mathbf{A}_k[j,l]$ is one if any agent in $\mathcal{B}_j$ at time instant k is able to transition to $\mathcal{B}_l$ by the next time instant, and zero otherwise. $\mathbf{A}_k$ is *symmetric* and *irreducible* (defined in Appendix); $\mathbf{A}_k[j,j] = 1$, $\forall j$. Equivalently, it can be also said that the topology of the bins is modelled as a bidirectional and strongly-connected graph, $\mathcal{G}_k = (\mathbf{A}_k, \mathcal{B})$, where $\mathbf{A}_k$ is edges (i.e., adjacent matrix) and $\mathcal{B}$ is nodes (i.e., bins).

Definition 3 (*Agent's state*)**.** Let $\mathbf{s}_k^i \in \{0,1\}^{n_b}$ be the state indicator vector of agent $a_i \in \mathcal{A}$ at time instant k. If the agent's state belongs to bin $\mathcal{B}_j$, then $\mathbf{s}_k^i[j] = 1$, otherwise 0. Note that the definition of the time instant will be described later in Definition 8.

Definition 4 (*Current swarm distribution*)**.** *The current (global) swarm distribution* $\mathbf{x}_k \in \mathbb{P}^{n_b}$ is a row-stochastic vector such that each element $\mathbf{x}_k[j]$ is the population

fraction (or *swarm density*) of $\mathcal{A}$ in bin $\mathcal{B}_j$ at time instant k:

$$\mathbf{x}_k := \frac{1}{|\mathcal{A}|} \sum_{\forall a_i \in \mathcal{A}} \mathbf{s}_k^i. \tag{2.1}$$

Definition 5 (*Stochastic policy*)**.** The probability that agent a_i in bin $\mathcal{B}_j$ at time instant k will transition to bin $\mathcal{B}_l$ before the next time instant is called its *stochastic policy*, denoted as:

$$M_k^i[j,l] := \text{Prob}(\mathbf{s}_{k+1}^i[l] = 1 | \mathbf{s}_k^i[j] = 1).$$

Note that $M_k^i \in \mathbb{P}^{n_b \times n_b}$ is a row-stochastic matrix such that $M_k^i \geq \mathbf{0}$ and $M_k^i \cdot \mathbf{1}^\top = \mathbf{1}^\top$, and will be referred as *Markov matrix*.

Assuming that all agents in bin $\mathcal{B}_j$ at time instant k are independently governed by an identical row-stochastic vector (denoted by $M_k[j,l]$, $\forall l$), it can be said that the ensemble of the swarm is evolved as

$$\mathbf{x}_{k+1} = \mathbf{x}_k M_k, \tag{2.2}$$

as n_a increases towards infinity. The underlying assumption is reasonably supported by Assumption 2, which will be shown later. Despite that, we will examine in Section 2.6.4 the performance degradation caused by possible differences between individual M_k^i on the proposed framework. Please keep in mind that, for each agent in $\mathcal{B}_j$, it is not necessary to know the other bins' stochastic policies (i.e., $M_k[j',l], \forall j' \neq j, \forall l$), and that this chapter only introduces such a matrix form, e.g., Equation (2.2), for the sake of theoretical analysis of the ensemble.

Every agent in each bin $\mathcal{B}_j$ executes the following algorithm at every time instant. The detail regarding how to generate its stochastic policies (i.e., Line 2) will be presented in Section 2.3.

Definition 6 (*Desired swarm distribution*)**.** *The desired swarm distribution* $\Theta \in \mathbb{P}^{n_b}$ is a row-stochastic vector such that each entry $\Theta[j]$ indicates the desired swarm density for bin $\mathcal{B}_j$.

Assumption 1. For ease of description for this chapter, we assume that $\Theta[j] > 0$, $\forall j \in \{1, ..., n_b\}$. In practice, there may exist some bins whose desired swarm densities

Algorithm 1 Probabilistic Swarm Distribution Guidance

// For each agent at time instant k:

1: Identify the current bin $\mathcal{B}_j$

2: Compute $M_k[j,l]$, $\forall l$;

3: Draw a random number z from the uniform distribution on an interval $[0,1]$

4: Choose bin $\mathcal{B}_q$ such that

$$\sum_{l=1}^{q-1} M_k[j,l] \leq z < \sum_{l=1}^{q} M_k[j,l]$$

5: Move to the selected bin

are zero. These bins can be accommodated by adopting any subroutine ensuring that all agents eventually move to and remain in any of the positive-desired-density bins (for example, please refer to Section 2.6.6). In this case, it should be assumed that the agent motion constraints over every bin $\mathcal{B}_j$ such that $\Theta[j] > 0$ are at least (bidirectionally) strongly-connected (i.e., $(\Theta^\top \Theta) \odot \mathbf{A}_k$ is irreducible).

Assumption 2 (*Communicational connectivity over bins*)**.** The physical motion constraint of a robotic agent is, in general, more stringent than its communicational constraint. From this, it can be assumed that if transition of agents between bin $\mathcal{B}_j$ and $\mathcal{B}_l$ is allowed within a unit time interval (i.e., $\mathbf{A}_k[j,l] = 1$), then one bin is within the communication range of agents in the other bin, and vice versa.

Definition 7 (*Neighbour bins, neighbour agents, and local information*)**.** For each bin $\mathcal{B}_j$, we define the set of its *neighbour bins* as $\mathcal{N}_k(j) = \{\forall \mathcal{B}_l \in \mathcal{B} \mid \mathbf{A}_k[j,l] = 1\}$. From Assumption 2, each agent in $\mathcal{B}_j$ can directly communicate with other agents in $\mathcal{N}_k(j)$. The set of these agents is called *neighbour agents*, denoted by $\mathcal{A}_{\mathcal{N}_k(j)} = \{\forall a_i \in \mathcal{A} \mid \mathbf{s}_k^i[l] = 1,\ \forall l : \mathcal{B}_l \in \mathcal{N}_k(j)\}$. This chapter refers to the local knowledge available from $\mathcal{A}_{\mathcal{N}_k(j)}$ as *the local information of the agents in bin* $\mathcal{B}_j$.

Assumption 3 (*Known Information*)**.** Each agent has reasonable sensing capabilities such that agents in bin $\mathcal{B}_j$ can perceive neighbour bins' information such as $\Theta[l]$ and $\mathbf{A}_k[j,l]$ in real time under Assumption 2. Note that the global information regarding Θ and $\mathbf{A}_k$ are not required to be known by the agents *a priori*, as will be described in Remark 3 later. Other pre-determined values such as variables regarding objective functions and design parameters (which will be introduced later) are known to all the agents.

Assumption 4 (*Agent's capability*)**.** Each agent can determine the bin to which it belongs, and know the locations of neighbour bins so that it can navigate towards any of the bins. Note that the dynamics of every agent is considered to be holonomic. The agent is capable of avoiding collision with other agents.

Assumption 5 (*The number of agents*)**.** The number of agents n_a is large enough such that the time evolution of the swarm distribution is governed by the Markov process in Equation (2.2). Although the finite cardinality of the agents may cause a residual convergence error (i.e., a sense of the difference between Θ and $\mathbf{x}_\infty$), a lower bound on n_a that probabilistically guarantees a certain level of convergence error is analysed in [10, Theorem 6]. Note that this theorem is generally appliable and thus is also valid for our work.

Definition 8 (*Time instant*)**.** We define *time instant* k to be the time when all the agents complete not only transitioning towards the bins selected at time instant $k-1$, but also obtain the local information necessary to construct M_k. Hence, a temporal interval (in the real-time scale) between any two sequential time instants may not always be consistent in practice. This might be because of the required inter-agent communication and/or physical congestion, which are varied at every time instant. In the worst case, due to some bins whose agents are somehow not ready in terms of transitioning or obtaining local knowledge, the temporal interval may be arbitrarily elongated. However, the proposed method can accommodate those bins by incorporating the asynchronous implementation in Section 2.5. It is worth mentioning that since our proposed approach demands relatively less communication burden on the agents, the temporal intervals would be diminished than those in GICA-based approaches.

2.3 The Proposed Closed-loop-type Framework under LICA

The objective of the swarm distribution guidance problem considered in this chapter is to distribute a set of agents $\mathcal{A}$ over a set of bins $\mathcal{B}$ by the Markov matrix M_k in a manner that holds the following desired features:

Desired Feature 1. The swarm distribution $\mathbf{x}_k$ asymptotically converges to the desired swarm distribution Θ as time instant k goes to infinity.

Desired Feature 2. Transition of the agents between the bins is controlled in a way that M_k becomes close to I as $\mathbf{x}_k$ converges to Θ. This implies that the agents are settling down as being close to Θ, and thus unnecessary transitions, which would have occurred in an open-loop-type framework, can be reduced. Moreover, the agents identify and compensate any partial loss or failure of the swarm distribution.

Desired Feature 3. For each agent in bin $\mathcal{B}_j$, the information required for generating time-varying stochastic policies is not global information about the entire agents $\mathcal{A}$ but only local information available from local agent group $\mathcal{A}_{\mathcal{N}_k(j)}$. Thereby, the resultant time-inhomogeneous Markov process is based on LICA, and has benefits such as reduced inter-agent communication, a shorter timescale for obtaining new information (than GICA), and the ability to be implemented asynchronously.

This section proposes a LICA-based framework for the swarm distribution guidance problem. The framework is different from the recent closed-loop-type algorithms in [8,10] in the sense that they utilise global information (e.g., the current swarm distribution in Equation (2.1)) to construct a time-inhomogeneous Markov matrix, whereas ours uses the local information in Equation (2.6), which will be shown later. We present that, in spite of using such relatively insufficient information, the desired features aforementioned can be achieved in the proposed framework. Before that, we first introduce the biological finding, which is about decision-making mechanisms of a fish swarm, that inspires this framework to particularly attain Desired Feature 3.

2.3.1 The Biological Inspiration

For a swarm of fish, it has commonly been assumed that their crowdedness limits their perception ranges over other members, and their cardinality restricts the capacity for individual recognition [21]. How fish end up with collective behaviours is different from the ways of other social species such as bees and ants, which are known to use recruitment signals for the guidance of the entire swarm [24, 25]. Thus, in the biology domain, a question naturally has arisen about the decision-making mechanism of fish

in an environment where local information is only available and information transfer between members does not explicitly happen [20–23, 26, 27].

It has been experimentally shown that fish's swimming activities vary depending on their perceivable neighbours. According to [26], fish have the tendency to maintain their statuses (e.g., position, speed, and heading angle) relative to those of other nearby fish, which results in their organised formation structures. In addition, it is presented in [27] that spatial density of fish has influences on both the minimum distances between them and the primary orientation of the fish school. Based on this knowledge, the works in [20–23] suggest individual-based models to further understand the collective behavioural mechanisms of fish: for example, their repelling, attracting, and orientating behaviours [20,22]; how the density of informed fish affects the elongation of the formation structure [21]; and group-size choices [23]. The common and fundamental characteristic of these models is that every agent maintains or adjusts its personal status with consideration of those of other individuals within its limited perception range.

As inspired by the understanding of fish, we believe that there must be an enhanced swarm distribution guidance approach in which each agent only needs to keep its relative status by relying on local information available from its nearby neighbours. In this approach, a global information is not necessary to be known by agents, and thereby the corresponding requirement of extensive information sharing over all the agents can be alleviated.

2.3.2 The Local Information required in the Proposed Approach

Overall, what we will show from now on is that global information is actually not required to generate feedback gains to operate closed-loop-type frameworks for robotic swarms. Instead, the main underlying idea is to use the deviation of current and desired swarm density at each local bin as its local feedback gain. Specifically, in most of GICA-based frameworks, the feedback gains are generated from the difference between $\mathbf{x}_k$ and Θ, which needs global information. Whereas, in the proposed LICA-based framework, agents in bin $\mathcal{B}_j$ use the difference between *the current local swarm density* $\bar{\mathbf{x}}_k[j]$ and

the *locally-desired swarm density* $\bar{\Theta}[j]$, which are respectively defined as follows:

$$\bar{\mathbf{x}}_k[j] := \frac{\mathbf{n}_k[j]}{\sum_{\forall \mathcal{B}_l \in \mathcal{N}_k(j)} \mathbf{n}_k[l]}, \tag{2.3}$$

where $\mathbf{n}_k[j]$ is the cardinality of agents in $\mathcal{B}_j$ at time instant k; and

$$\bar{\Theta}[j] := \frac{\Theta[j]}{\sum_{\forall \mathcal{B}_l \in \mathcal{N}_k(j)} \Theta[l]}. \tag{2.4}$$

It turns out that the two values are both locally-available information within $\mathcal{A}_{\mathcal{N}_k(j)}$. The difference between $\bar{\mathbf{x}}_k[j]$ and $\bar{\Theta}[j]$ is utilised for a local-information-based feedback gain, denoted by $\bar{\xi}_k[j]$, which should be a scalar in $(0, 1]$ that monotonically decreases as $\bar{\mathbf{x}}_k[j]$ converges to $\bar{\Theta}[j]$. For instance, this chapter uses

$$\bar{\xi}_k[j] := \left(\frac{\left| \bar{\Theta}[j] - \bar{\mathbf{x}}_k[j] \right|}{\bar{\Theta}[j]} \right)^{\alpha}, \tag{2.5}$$

being saturated to $[\epsilon_\xi, 1]$ if the value lies outside this range, where $\alpha > 0$ and $\epsilon_\xi > 0$ are design parameters. This gain is called *primary local-feedback gain*, controling the primary guidance matrix P_k (shown in the next subsection). Here, $\alpha > 0$ is the sensitivity parameter affecting $\bar{\xi}_k[j]$ with regard to the difference between $\bar{\mathbf{x}}_k[j]$ and $\bar{\Theta}[j]$ (as shown in Figure 2.5(a)); $\epsilon_\xi > 0$ is a reasonably small positive value ensuring that always $\bar{\xi}_k[j] > 0$ in order to mathematically guarantee (R4), which will be described in the next subsection. How to use $\bar{\xi}_k[j]$ explicitly may be different depending on different applications, and hence it will be given along with some implementation examples in Section 2.4.

Remark 1. Equation (2.3) is equivalent to the j-th element of the following vector:

$$\bar{\mathbf{x}}_k^j := \frac{1}{|\mathcal{A}_{\mathcal{N}_k(j)}|} \sum_{\forall a_i \in \mathcal{A}_{\mathcal{N}_k(j)}} \mathbf{s}_k^i. \tag{2.6}$$

Here, we intentionally introduce Equation (2.6) for ease of comparison with the information required in the existing literature (i.e., Equation (2.1)). Equation (2.6) implies that, in order for each agent in bin $\mathcal{B}_j$ to estimate $\bar{\mathbf{x}}_k^j[j]$ (i.e., the current local swarm density $\bar{\mathbf{x}}_k[j]$), the set of other agents whose information is necessary is just $\mathcal{A}_{\mathcal{N}_k(j)}$. That is, each agent needs to have neither a large perception radius nor an extensive information consensus process over the entire agents.

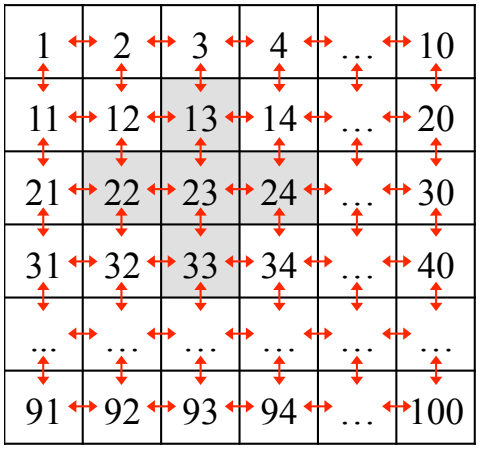


Figure 2.2: An example showing how to calculate $\bar{\mathbf{x}}_k[j]$: for bin $\mathcal{B}_{23}$, $\bar{\mathbf{x}}_k[23] = \mathbf{n}_k[23]/(\mathbf{n}_k[13]+\mathbf{n}_k[22]+\mathbf{n}_k[23]+\mathbf{n}_k[24]+\mathbf{n}_k[33])$. In the proposed framework, agents in the bin only need to obtain the local information from other agents in its neighbour bins (shaded). Note that each square indicates each bin, and the red arrow between two bins $\mathcal{B}_j$ and $\mathcal{B}_l$ means that $\mathbf{A}_k[j,l] = 1$.

Remark 2. In the rest of this chapter, it is assumed that the current local swarm density $\bar{\mathbf{x}}_k[j]$ in Equation (2.3) is accurately accessible by each agent in bin $\mathcal{B}_j$, for the ease of description. This can actually happen via a simple multi-hop communication over all the agents in $\mathcal{A}_{\mathcal{N}_k(j)}$. In order to reduce the required communication burden, we could utilise distributed density estimation methods in [8, 28] at the expense of a certain level of estimation error. Hence, we will numerically examine the effect of the uncertainty on the proposed framework, as will be shown in Section 2.6.4.

Remark 3. It is worth repeating that, in the proposed approach, each agent in bin $\mathcal{B}_j$ only relies on its local information about its neighbour bins $\mathcal{N}_k(j)$. This also applies to mission information such as the desired distribution Θ. That is, the agent does not necessarily need to know the entire desired distribution *a priori* (which is the case in most of the existing works), but can obtain $\bar{\Theta}[j]$ in a real-time manner during a mission as long as Assumption 2 holds. This is also the case for the motion constraint $\mathbf{A}_k$ as long as motion constraints regarding neighbour bins are perceivable under Assumption 2 along with reasonably capable sensors

2.3.3 The LICA-based Markov matrix

This subsection presents our methodology to generate a time-inhomogeneous Markov matrix M_k that achieves Desired Features 1–3 by using the local information feedback. The basic form of the stochastic policy for every agent in bin $\mathcal{B}_j$ is such that

$$M_k[j,l] := (1-\omega_k[j])P_k[j,l] + \omega_k[j]S_k[j,l], \quad \forall \mathcal{B}_l \in \mathcal{B}. \tag{2.7}$$

Here, $\omega_k[j] \in [0,1)$ is the weighting factor to have different weights on the *primary policy* $P_k[j,l] \in \mathbb{P}$ and the *secondary policy* $S_k[j,l] \in \mathbb{P}$. It is defined as

$$\omega_k[j] := \exp(-\lambda k) \cdot G_k[j] \tag{2.8}$$

where λ is a design parameter that controls decay of $\omega_k[j]$; and $G_k[j] \in [0,1]$ is *secondary local-feedback gain*, which activates $S_k[j,l]$ depending on the difference between $\bar{\mathbf{x}}_k[j]$ and $\bar{\Theta}[j]$. Note that $\omega_k[j]$ is mainly affected by $G_k[j]$, while diminishing as time instant k goes to infinity.

We introduced two policies P_k and S_k in order to help prospective users to have more design flexibility when implementing the framework into their own specific problems. As you will see, the asymptotic stability of agents towards Θ is in fact guaranteed by P_k under the condition that the following Requirements 1–4 are satisfied. Since $\omega_k[j]$ is diminishing as time instant k goes to infinity, users may adopt any temporary policies as S_k in addition to P_k, if necessary. For instance, when it is desired to disperse agents in $\mathcal{B}_j$ into its neighbour bins more quickly if the bin is too overpopulated, it can happen by setting $S_k[j,l] = 1/|\mathcal{N}_k(j)|, \forall \mathcal{B}_l \in \mathcal{N}_k(j)$ and that $G_k[j]$ is designed to be close to one when $\bar{\mathbf{x}}_k[j] \gg \bar{\Theta}[j]$, zero otherwise. Note that Section 2.4 will give explicit descriptions of P_k, S_k, and G_k, which are varied depending on specific implementations.

Equation (2.7) can be represented in matrix form as

$$M_k = (I - W_k)P_k + W_k S_k, \tag{2.9}$$

where $P_k \in \mathbb{P}^{n_b \times n_b}$ and $S_k \in \mathbb{P}^{n_b \times n_b}$ are row-stochastic matrices, called *primary guidance matrix* and *secondary guidance matrix*, respectively. $W_k \in \mathbb{R}^{n_b \times n_b}$ is a diagonal matrix such that $W_k = \mathrm{diag}(\omega_k[1], ..., \omega_k[n_b])$.

We claim that, in order for the Markov system to achieve all the desired Features, P_k must satisfy the following requirements.

Requirement 1. P_k is a matrix with row sums equal to one, i.e.,

$$\sum_{l=1}^{n_b} P_k[j,l] = 1, \quad \forall j. \tag{R1}$$

In fact, P_k needs to be row-stochastic, for which it should further hold that $P_k[j,l] \geq 0$, $\forall j, l$. Note that this constraint is implied by (R4), which will be introduced later.

Requirement 2. All diagonal elements are positive, i.e.,

$$P_k[j,j] > 0, \quad \forall j. \tag{R2}$$

Requirement 3. The stationary distribution (i.e., equilibrium) of P_k is the desired swarm distribution Θ, i.e., $\Theta^\top P_k = \Theta^\top$ (or $\sum_{j=1}^{n_b} \Theta[j]P_k[j,l] = \Theta[l], \forall l$). Along with (R1), this can be fulfilled by setting

$$\Theta[j]P_k[j,l] = \Theta[l]P_k[l,j], \quad \forall j, \forall l. \tag{R3}$$

A Markov process satisfying this property is said to be *reversible*.

Requirement 4. P_k is irreducible such that

$$\begin{aligned} P_k[j,l] &> 0 \quad \text{if } \mathbf{A}_k[j,l] = 1. \\ P_k[j,l] &= 0 \quad \text{otherwise.} \end{aligned} \tag{R4}$$

Note that $\mathbf{A}_k$ is also irreducible from in Definition 2.

Requirement 5. P_k becomes close to I as $\bar{\mathbf{x}}_k$ converges to $\bar{\Theta}$, i.e.,

$$P_k[j,j] \to 1 \text{ as } \bar{\mathbf{x}}_k[j] \to \bar{\Theta}[j] \text{ (or } \bar{\xi}_k[j] \to 0), \quad \forall j. \tag{R5}$$

Every agent in each bin $\mathcal{B}_j$ executes the following subroutine to generate its stochastic policies at every time instant. Depending on missions, $\bar{\xi}_k[j]$, P_k, S_k, and $G_k[j]$ can be designed differently under given specific constraints (the detail regarding Lines 4-6 will be presented in Section 2.4, which shows examples of how to implement this framework). As long as P_k holds (R1)-(R5) for every time instant k, the aforementioned desired features are achieved. Note that (R1)-(R4) are associated with Desired Feature 1, whereas (R5) is with Desired Features 2 and 3. The detailed analysis will be described in the next subsection.

In a nutshell, our design guidelines are as follows:

Algorithm 2 Generation of $M_k[j,l], \forall l$ (Line 2 in Algorithm 1)

// Obtain the local information

1: Compute $\bar{\Theta}[j]$ using (2.4);

2: Obtain $\bar{\mathbf{x}}_k[j]$;

// Generate stochastic policies

3: Compute $\bar{\xi}_k[j]$ (using (2.5));

4: Compute $P_k[j,l]$, $\forall l$;

5: Compute $S_k[j,l]$, $\forall l$;

6: Compute $G_k[j]$;

7: Compute $\omega_k[j]$ using (2.8);

8: Compute $M_k[j,l]$ using (2.7), $\forall l$;

i. Design $\bar{\xi}_k[j]$ as a scalar function in $(0,1]$ that monotonically decreases as $\bar{\mathbf{x}}_k[j]$ converges to $\bar{\Theta}[j]$, e.g., Equation (2.5). Note that the shape of $\bar{\xi}_k[j]$ is important so that it may cause high residual convergence error, as will be shown in Section 2.6.1.

ii. Design $P_k[j,l]$ that satisfies (R1)–(R5) along with additional criteria from a given specific application.

iii. Design $S_k[j,l]$ with consideration of the robotic swarm's desired temporary behaviours that help the ultimate problem objective (if necessary).

iv. Design $G_k[j]$ as a scalar function in $[0,1]$ in terms of $\bar{\mathbf{x}}_k[j]$ and $\bar{\Theta}[j]$ (e.g., Equations (2.17) or (2.27)), with consideration of when S_k is desired to be activated (if S_k is implemented).

v. Use $M_k[j,l]$ and $\omega_k[j]$ as shown in Equations (2.7) and (2.8), respectively.

We will apply the same guidelines when implementing the proposed framework into the specific examples in Section 2.4.

2.3.4 Analysis

We first show that the Markov process using Equation (2.9) holds Desired Feature 1 under the assumption that P_k satisfies the requirements (R1)-(R4) for every time instant. The swarm distribution at time instant $k \geq k_0$, governed by the Markov process from an arbitrary initial state $\mathbf{x}_{k_0}$, can be written as:

$$\mathbf{x}_k = \mathbf{x}_{k_0} U_{k_0,k} := \mathbf{x}_{k_0} M_{k_0} M_{k_0+1} \cdots M_{k-1}. \tag{2.10}$$

Theorem 1. *Provided that the requirements (R1)-(R4) are satisfied for all time instants $k \geq k_0$, it holds that $\lim_{k\to\infty} \mathbf{x}_k = \Theta$ pointwise for all agents, irrespective of the initial condition.*

Proof. Please refer to Appendix A. □

Theorem 1 implies that the ensemble of the agents eventually converges to the desired swarm distribution, regardless of S_k, $G_k[j]$, and (R5). However, the system may induce unnecessary transitions of agents even after being close enough to Θ, meaning that Desired Feature 2 does not hold yet.

From now on, we will present that Desired Features 2 and 3 can also be obtained if the requirement (R5) is additionally satisfied. Suppose that, for every bin $\mathcal{B}_j$, $\bar{\mathbf{x}}_k[j]$ converges to and eventually reaches $\bar{\Theta}[j]$ at some time instant k. From Equations (2.3)-(2.4) and the supposition of $\bar{\mathbf{x}}_k[j] = \bar{\Theta}[j]$, $\forall j$, it follows that $1/\bar{\Theta}[j] \cdot \mathbf{n}_k[j] = \sum_{\forall \mathcal{B}_l \in \mathcal{N}_k(j)} \mathbf{n}_k[l]$, $\forall j$. This can be represented in matrix form as:

$$\mathbf{n}_k \cdot B := \mathbf{n}_k \cdot (\mathbf{A}_k - X) = \mathbf{0} \tag{2.11}$$

where $X \in \mathbb{R}^{n_b \times n_b}$ is a diagonal matrix such that $X = \text{diag}(1/\bar{\Theta}[1], ..., 1/\bar{\Theta}[n_b])$.

Lemma 1. *Let the term* tree-type (bidirectional) topology *refer to as a graph such that any two vertices are connected by exactly one bidirectional path with no cycles (e.g., Figure 2.3(a)). Given n_b bins connected as a tree-type topology, the rank of its corresponding matrix B in Equation (2.11) is $n_b - 1$.*

Proof. The matrix $B \in \mathbb{R}^{n_b \times n_b}$ can be linearly decomposed into n_e of the same-sized matrices $B_{(i,j)}$, where n_e is the number of edges in the underlying graph of $\mathbf{A}_k$.

(a) Tree-type **(b)** Arbitrarily connected

Figure 2.3: Examples of simple bin topologies to help Lemma 1 & 2: (a) tree-type; (b) arbitrarily connected. The red line in (b) indicates a newly-added route between bin $\mathcal{B}_1$ and $\mathcal{B}_4$ based on the topology in (a).

Here, $B_{(i,j)} \in \mathbb{R}^{n_b \times n_b}$ is a matrix such that $B_{(i,j)}[i,i] = -\Theta[j]/\Theta[i]$ and $B_{(i,j)}[j,j] = -\Theta[i]/\Theta[j]$; $B_{(i,j)}[i,j] = B_{(i,j)}[j,i] = 1$; and all the other entries are zero. For example, consider that four bins are given and connected as shown in Figure 2.3(a). Clearly, $B = B_{(1,2)} + B_{(2,3)} + B_{(2,4)}$, where

$$B = \begin{pmatrix} -\frac{\Theta[2]}{\Theta[1]} & 1 & 0 & 0 \\ 1 & -\frac{\Theta[1]+\Theta[3]+\Theta[4]}{\Theta[2]} & 1 & 1 \\ 0 & 1 & -\frac{\Theta[2]}{\Theta[3]} & 0 \\ 0 & 1 & 0 & -\frac{\Theta[2]}{\Theta[4]} \end{pmatrix},$$

$$B_{(1,2)} = \begin{pmatrix} -\frac{\Theta[2]}{\Theta[1]} & 1 & 0 & 0 \\ 1 & -\frac{\Theta[1]}{\Theta[2]} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$B_{(2,3)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -\frac{\Theta[3]}{\Theta[2]} & 1 & 0 \\ 0 & 1 & -\frac{\Theta[2]}{\Theta[3]} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$B_{(2,4)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -\frac{\Theta[4]}{\Theta[2]} & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -\frac{\Theta[2]}{\Theta[4]} \end{pmatrix}.$$

It turns out that the rank of every $B_{(i,j)}$ is one, and the matrix has only one linearly independent column vector, denoted by $v_{(i,j)}$. Without loss of generality, we consider $v_{(i,j)} \in \mathbb{R}^{n_b}$ as a column vector such that the i-th entry is $-\frac{1}{\Theta[i]}$, the j-th entry is $\frac{1}{\Theta[j]}$, and the others are zero: for an instance, $v_{(1,2)} = [-\frac{1}{\Theta[1]}, \frac{1}{\Theta[2]}, 0, 0]^\top$.

It follows that $v_{(i,j)}$ and $v_{(k,l)}$ are linearly independent when the bin pairs $\{i, j\}$ and $\{k, l\}$ are different. This implies that the number of linearly independent column vectors of B is the same as that of edges in the topology. Hence, for a tree-type topology of n_b bins, since there exist $n_b - 1$ edges, the rank of the corresponding matrix B is $n_b - 1$. □

Lemma 2. *Given a bidirectional and strongly-connected topology of bins, the rank of its corresponding matrix B is not affected by adding a new bidirectional edge that directly connects any two existing bins.*

Proof. We will show that this claim is valid even when a tree-type topology is given, as it is a sufficient condition for being bidirectional and strongly-connected. Given the tree-type topology in Figure 2.3(a), suppose that bin $\mathcal{B}_1$ and $\mathcal{B}_4$ are newly connected. Then, the new topology becomes as shown in Figure 2.3(b), and it has a new corresponding matrix B_{new}, where $B_{new} = B + B_{(1,4)}$. As explained in the proof of Lemma 1, the rank of $B_{(1,4)}$ is one and it has only a linearly independent vector $v_{(1,4)}$. However, this vector can be produced as a linear combination of the existing v vectors of B (i.e., $v_{(1,4)} = v_{(1,2)} + v_{(2,4)}$). Thus, the rank of B_{new} retains that of B. Without loss of generality, this implies that the rank of B of a given bidirectional and strongly-connected topology is not affected by adding a new edge that directly connects any two existing bins. □

Thanks to Lemma 1 and 2, we end up with the following corollary and lemma:

Corollary 1. *Given n_b bins that are at least bidirectional and strongly-connected, the rank of its corresponding B is $n_b - 1$.*

Lemma 3. *Given n_b bins that are at least bidirectional and strongly-connected, convergence of $\bar{\mathbf{x}}_\infty$ to $\bar{\Theta}$ is equivalent to convergence of $\mathbf{x}_\infty$ to Θ.*

Proof. Assuming that $\lim_{k\to\infty} \bar{\mathbf{x}}_k = \bar{\Theta}$, it can be said that $\lim_{k\to\infty} \mathbf{n}_k \cdot B = \mathbf{0}$, as similar to the derivation of Equation (2.11). From Equation (2.4), it turns out that

$$\Theta \cdot B = \mathbf{0}. \tag{2.12}$$

Since the nullity of B is one due to Corollary 1, there exists only one linearly-independent row-vector $\mathbf{a} \in \mathbb{R}^{n_b}$ such that $\mathbf{a} \cdot B = \mathbf{0}$. Hence, it follows that $\lim_{k\to\infty} \mathbf{n_k} = \epsilon\Theta$, where ϵ is an arbitrary scalar value. This also implies that $\lim_{k\to\infty} \mathbf{x}_k = \lim_{k\to\infty} \mathbf{n}_k / n_a = \Theta$.

On the other hand, supposing that $\lim_{k\to\infty} \mathbf{x}_k = \Theta$, it can be rewritten as $\lim_{k\to\infty} \mathbf{n}_k = n_a\Theta$. By right multiplying B for both side (i.e., $\lim_{k\to\infty} \mathbf{n}_k \cdot B = n_a(\Theta \cdot B)$), it follows from Equation (2.12) that $\lim_{k\to\infty} \mathbf{n}_k \cdot B = \mathbf{0}$. By the definition of B, it can be rearranged as $\lim_{k\to\infty} \bar{\mathbf{x}}_k = \bar{\Theta}$.

Therefore, convergence of $\bar{\mathbf{x}}_\infty$ to $\bar{\Theta}$ is equivalent to convergence of $\mathbf{x}_\infty$ to Θ. □

From this lemma and (R5), Desired Feature 2 finally holds as follows.

Theorem 2. *If P_k satisfies (R1)–(R5) for all time instants $k \geq k_0$, the Markov process using M_k in Equation (2.9) satisfies Desired Feature 2 as well as Desired Feature 1.*

Proof. It was shown from Theorem 1 that the requirements (R1)–(R4) guarantee the convergence of $\mathbf{x}_\infty$ to Θ (i.e., Desired Feature 1). From this and Lemma 3, $\bar{\mathbf{x}}_\infty$ also converges to $\bar{\Theta}$. If P_k additionally complies with (R5), then P_k becomes close to I as $k \to \infty$. This is also the case for the Markov process M_k, which satisfies Desired Feature 2. □

Corollary 2. *In order for every agent in bin $\mathcal{B}_j$ to generate $M_k[j, l], \forall l$ in Equation (2.7), the agent only needs local information within $\mathcal{A}_{\mathcal{N}_k(j)}$. Therefore, Desired Feature 3 is also achieved.*

Remark 4 (*Robustness against dynamic changes of agents or bins*)**.** The proposed framework is robust against dynamic changes in the number of agents or bins. As each agent behaves based on its current bin location and local information in a memoryless manner, Desired Features 1–3 in the proposed framework will not be affected by inclusion or exclusion of agents in a swarm.

Besides, as long as changes on bins are perceived by nearby agents in the corresponding neighbour bins, robustness against those changes also holds in the proposed framework. This is because agents in bin $\mathcal{B}_j$ utilise only local information such as $\bar{\Theta}[j]$ and $\bar{\mathbf{x}}_k[j]$, and are not required to know information from other far-away bins. Moreover, the proposed framework does not need to recalculate Θ (which has to be normalised in a GICA-based framework such that $\sum_{\forall j} \Theta[j] = 1$ after reflecting such changes) because computing $\bar{\Theta}[j]$ in Equation (2.4) already includes a sense of normalisation based on local information.

2.4 Implementation Examples

2.4.1 Example I: Minimising Travelling Expenses

This section provides implementation examples of the proposed framework. In particular, this subsection addresses a problem of minimising travelling expenses of agents during convergence to a desired swarm distribution, as shown in [10]. The problem can be defined as follows:

Problem 1. Given a cost matrix $E_k \in \mathbb{R}^{n_b \times n_b}$ in which each element $E_k[j,l]$ represents the travelling expense of an agent from bin $\mathcal{B}_j$ to $\mathcal{B}_l$, find P_k such that

$$\min \sum_{j=1}^{n_b} \sum_{l=1}^{n_b} E_k[j,l] P_k[j,l], \tag{P1}$$

subject to (R1)-(R5) and

$$\epsilon_M \Theta[l] f_\xi(\bar{\xi}_k[j], \bar{\xi}_k[l]) f_E(E_k[j,l]) \leq P_k[j,l] \quad \text{if } \mathbf{A}_k[j,l] = 1, \forall j \neq l, \tag{2.13}$$

where $\Theta[l]$ enables agents in bin $\mathcal{B}_j$ to be distributed over its neighbour bins in proportion to the desired swarm distribution; $f_\xi(\bar{\xi}_k[j], \bar{\xi}_k[l]) \in (0,1]$ and $f_E(E_k[j,l]) \in (0,1]$

basically control the lower bound of $P_k[j,l]$ in Equation (2.13), depending on the primary local-feedback gains and travelling expenses, respectively. Specifically, it is set that

$$f_\xi(\bar{\xi}_k[j], \bar{\xi}_k[l]) = \max(\bar{\xi}_k[j], \bar{\xi}_k[l]) \tag{2.14}$$

so that the value monotonically increases with regard to increase of either $\bar{\xi}_k[j]$ or $\bar{\xi}_k[l]$ and diminishes as $\bar{\xi}_k[j]$ and $\bar{\xi}_k[l]$ simultaneously reduce, meaning that it allows a larger number of transitioning agents between the two bins $\mathcal{B}_j$ and $\mathcal{B}_l$ when any one of them needs to be regulated. $f_E(E_k[j,l]) \in (0,1]$ monotonically decreases as $E_k[j,l]$ increases (see Equation (2.29) for an exemplar of its explicit definition), preventing agents in bin $\mathcal{B}_j$ from spending higher transition expenses. We assume that E_k is symmetric; $E_k[j,l] > 0$ if $\mathbf{A}_k[j,l] = 1$; and its diagonal entries are zero.

Corollary 3. *The optimal matrix P_k of the problem (P1) is given by: $\forall j, l \in \{1, ..., n_b\}$ and $l \neq j$,*

$$P_k[j,l] = \begin{cases} \epsilon_M \Theta[l] f_\xi(\bar{\xi}_k[j], \bar{\xi}_k[l]) f_E(E_k[j,l]) & \text{if } \mathbf{A}_k[j,l] = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

and $\forall j$,

$$P_k[j,j] = 1 - \sum_{\forall l \neq j} P_k[j,l]. \tag{2.16}$$

Proof. Please refer to Appendix B in this chapter. □

To reduce unnecessary transitions of agents during this process, it is desirable that agents in bin $\mathcal{B}_j$ such that $\bar{\mathbf{x}}_k[j] \leq \bar{\Theta}[j]$ (i.e., underpopulated) do not deviate. To this end, we set $S_k = I$ and $G_k[j]$ as follows:

$$G_k[j] := \frac{\exp(\beta(\bar{\Theta}[j] - \bar{\mathbf{x}}_k[j]))}{\exp(\beta|\bar{\Theta}[j] - \bar{\mathbf{x}}_k[j]|)}. \tag{2.17}$$

The gain value is depicted in Figure 2.4(a) with regard to the sensitivity parameter β, which controls the steepness of $G_k[j]$ at around when $\bar{\mathbf{x}}_k[j] - \bar{\Theta}[j]$ is close to zero but positive. For example, at a lower β, a relatively higher number of agents tend to follow the secondary guidance matrix S_k (i.e., not to deviate) rather than P_k even when $\bar{\mathbf{x}}_k[j] - \bar{\Theta}[j] > 0$ is not much close to zero.

Remark 5 (*Increase of Convergence Rate*)**.** Due to the fact that $\sum_{\forall l \neq j} P_k[j,l] \leq \sum_{\forall \mathcal{B}_l \in \mathcal{N}_k(j) \setminus \{\mathcal{B}_j\}} \Theta[l]$ from Equation (2.15), the total outflux of agents from bin $\mathcal{B}_j$ becomes smaller as the bin has fewer connections with other bins. This eventually makes the convergence rate of the Markov process slower.

Adding an additional variable into $P_k[j,l]$ in (2.15) does not affect the obtainment of Desired Features 1-3 as long as P_k satisfies (R1)-(R5). Thus, in order to enhance the convergence rate under the requirements, one can add

$$\epsilon_\Theta := \min \left\{ \frac{1}{\sum_{\forall s: \mathcal{B}_s \in \mathcal{N}_k(j) \setminus \{\mathcal{B}_j\}} \Theta[s]}, \frac{1}{\sum_{\forall s: \mathcal{B}_s \in \mathcal{N}_k(l) \setminus \{\mathcal{B}_l\}} \Theta[s]} \right\} \tag{2.18}$$

into $P_k[j,l]$, as follows:

$$P_k[j,l] = \begin{cases} \epsilon_\Theta \epsilon_M \Theta[l] f_\xi(\bar{\xi}_k[j], \bar{\xi}_k[l]) f_E(E_k[j,l]) & \text{if } \mathbf{A}_k[j,l] = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.19}$$

which can be substituted for Equation (2.15).

Algorithm 3 Minimising Travelling Expenses (Lines 4–6 of Algorithm 2 for P1)

1: Compute $P_k[j,l]$ $\forall l$ using (2.15) (or (2.19)) and (2.16)
2: Set $S_k[j,j] = 1$; $S_k[j,l] = 0, \forall l \neq j$
3: Compute $G_k[j]$ using (2.17)

2.4.2 Example II: Maximising Convergence Rate within Flux Upper Limits

This subsection presents an example in which the specific objective is to maximise the convergence rate under upper bounds regarding transition of agents between bins, referred to as *flux upper limits*. The bounds can be interpreted as safety constraints in terms of collision avoidance and congestion: higher congestions may induce higher collisions amongst agents, which may bring unfavourable effects on system performance. A similar problem is addressed by an open-loop-type algorithm in [18], where transitions

of agents are limited only after a desired swarm distribution is achieved. This restriction is not for considering the aforementioned safety constraints, but rather for mitigating the trade-off between convergence rate and long-term system efficiency.

For the sake of imposing flux upper limits *during the entire process*, we consider the following one-way flux constraint: for every time instant k,

$$\mathbf{n}_k[j]P_k[j,l] \leq c_{(j,l)}, \quad \forall j, \forall l \neq j. \tag{2.20}$$

This means that the number of agents moving from bin $\mathcal{B}_j$ to $\mathcal{B}_l$ is upper-bounded by $c_{(j,l)}$. The bound value is assumed to be very small with consideration of mission environments such as the number of agents, the number of bins, and their topology. Otherwise, all the agents can be distributed over the bins very soon so that the flux upper limits become meaningless and the corresponding problem can be trivial. Please note that the flux limits in this example should be considered as expected constraints. In the case where hard constraints are to be accommodated in practice, it is necessary to set a tighter value with consideration of a margin from the actual value. The level of the margin would be affected by the number of agents involved in the framework, as will be shown in Figure 9(c) later.

Regarding the convergence rate of a Markov chain, there are respective analytical methods depending on whether it is time-homogeneous or time-inhomogeneous. For a time-homogeneous Markov chain, if the matrix is irreducible, the second largest eigenvalue of the matrix is used as an index indicating its asymptotic convergence rate [29, p.389]. In contrast, for a time-inhomogeneous Markov chain, *coefficients of ergodicity* can be utilised as a substitute for the second largest eigenvalue, which is not useful for this case [30]. Particularly, we use the following *proper* coefficient of ergodicity, amongst others:

Definition 9. (*Coefficient of Ergodicity* [31, pp. 136–137]). Given a stochastic matrix $\mathcal{M} \in \mathbb{P}^{n\times n}$, a (proper) coefficient of ergodicity $0 \leq \tau(\mathcal{M}) \leq 1$ can be defined as:

$$\tau(\mathcal{M}) := \max_{\forall s} \max_{\forall j, \forall l} \left| \mathcal{M}[j,s] - \mathcal{M}[l,s] \right|. \tag{2.21}$$

A coefficient of ergodicity is said to be *proper* if $\tau(\mathcal{M}) = 0$ is necessary and sufficient for $\mathcal{M} = \mathbf{1}^\top \cdot v$, where $v \in \mathbb{P}^n$ is a row-stochastic vector.

The convergence rate of a time-inhomogeneous Markov chain $\mathcal{M}_k \in \mathbb{P}^{n\times n}$, $\forall k \geq 1$ can be maximised by minimising $\tau(\mathcal{M}_k)$ at each time instant k, thanks to [31, Theorem 4.8, p.137]: $\tau(\mathcal{M}_1\mathcal{M}_2\cdots\mathcal{M}_r) \leq \prod_{k=1}^{r}\tau(\mathcal{M}_k)$. Hence, the objective of the specific problem considered in this subsection can be defined as: find P_k such that

$$\min \tau(P_k) \tag{2.22}$$

subject to (R1)-(R5) and (2.20).

Remark 6 (*Advantages of the coefficient of ergodicity in (2.21)*)**.** Other proper coefficients in [31, p. 137] such as

$$\tau_1(\mathcal{M}) = 1 - \min_{\forall j,\forall l}\sum_{\forall s}\min\{\mathcal{M}[j,s],\mathcal{M}[l,s]\}$$

or

$$\tau_2(\mathcal{M}) = 1 - \sum_{\forall s}\min_{\forall j}\{\mathcal{M}[j,s]\}$$

may have the trivial case such that $\tau_1(P_k) = 1$ (or $\tau_2(P_k) = 1$) for some time instant k, when they are applied to this problem. For example, given a topology of bins $\mathbf{A}_k$, there may exist a pair of bins $\mathcal{B}_j$ and $\mathcal{B}_l$ such that $P_k[j,s] = 0$ or $P_k[l,s] = 0$, $\forall s$. To avoid this trivial case, the work in [10] instead utilises $\tau_1((P_k)^{d_{\mathbf{A}_k}})$ as the proper coefficient of ergodicity, where $d_{\mathbf{A}_k}$ denotes the diameter of the underlying graph of $\mathbf{A}_k$. However, this implies that agents in bin $\mathcal{B}_j$ are required to additionally access the information from other bins beside $\mathcal{N}_k(j)$, causing additional communicational costs. The coefficient of ergodicity in (2.21) does not suffer from this issue. Note that $\tau(\mathcal{M}) \leq \tau_1(\mathcal{M}) \leq \tau_2(\mathcal{M})$ [31, p. 137].

Finding the optimal solution for the problem (2.22) is another challenging issue, called *fastest mixing Markov chain problem*. Since the purpose of this section is to show an example of how to implement our proposed framework, we heuristically address this problem at this moment.

Suppose that a matrix P_k satisfying (R1)-(R5) is given and that the topology of bins is at least connected without any bin being connected to all the others. Since the matrix is non-negative and there exists at least one zero-value entry in each column, the coefficient of ergodicity can be said as $\tau(P_k) = \max_{\forall s,\forall j}(P_k[j,s])$. Assuming that

$\max_{\forall l \neq j} P_k[j,l] \leq 1/|\mathcal{N}_k(j)|$, which is generally true due to the small values of $c_{(j,l)}$, it turns out that each diagonal element of P_k is the largest value in each row. Thus, we can say that $\tau(P_k) = \max_{\forall j} P_k[j,j]$. Eventually, the objective function in Equation (2.22) can be rewritten as $\text{maxmin}_{\forall j} \sum_{\forall l \neq j} P_k[j,l]$ because minimising the maximum diagonal element of a stochastic matrix is equivalent to maximising the minimum row-sum of its off-diagonal elements.

We turn now to the constraints (R1)-(R5) and (2.20). In order to comply with (R3), we initially set $P_k[j,l] = \Theta[l]Q_k[j,l]$, where Q_k is a symmetric matrix that we will design now. The constraint (2.20), (R4), and the symmetricity of Q_k are necessary conditions for the following constraint: $\forall j$, $\forall l \neq j$,

$$\begin{aligned} \min\left(\frac{c_{(j,l)}}{\mathbf{n}_k[j]\Theta[l]}, \frac{c_{(l,j)}}{\mathbf{n}_k[l]\Theta[j]}\right) \geq Q_k[j,l] > 0 \quad & \text{if } \mathbf{A}_k[j,l] = 1 \\ Q_k[j,l] = 0 \quad & \text{otherwise.} \end{aligned} \tag{2.23}$$

For (R2) and (R5), we set the diagonal entries of P_k as

$$P_k[j,j] \geq 1 - \bar{\xi}_k[j], \quad \forall j. \tag{2.24}$$

Note that the non-strict inequality is not troublesome to (R2) because $\bar{\xi}_k[j] = 1$ only when $\bar{\mathbf{x}}_k[j] = 0$, in which there exists no agent in bin $\mathcal{B}_j$, and thus effectively $P_k[j,j] = 1$. Equation (2.24) can be rewritten, with consideration of (R1) (i.e., $\sum_{l=1}^{n_b} \Theta[l]Q_k[j,l] = 1, \forall j$), as

$$\sum_{\forall l \neq j} \Theta[l]Q_k[j,l] \leq \bar{\xi}_k[j], \quad \forall j. \tag{2.25}$$

In summary, Equation (2.23) is a sufficient condition for (R3), (R4), and (2.20); and Equation (2.25) is for (R1), (R2), and (R5). Hence, the reduced problem can be defined as:

Problem 2. Find Q_k such that

$$\text{maxmin}_{\forall j} \sum_{\forall l \neq j} \Theta[l]Q_k[j,l] \tag{P2}$$

subject to (2.23) and (2.25).

The algorithm for the problem (P2) is shown in Algorithm 4. If we neglect (2.25), an optimal solution can be obtained by making $Q_k[j,l]$ equal to its upper bound in (2.23) (Line 2). However, this solution may not hold (2.25). Thus, we lower the entries of Q_k to satisfy (2.25), while keeping them symmetric and as high as possible (Lines 3–8). In detail, Line 3 (or Line 6) ensures the constraint (2.25) for each bin $\mathcal{B}_j$ in a way that, if this is not the case, obtains the necessary lowering factor $\bar{\epsilon}'_Q[j]$ (or $\bar{\epsilon}_Q[j]$). In order to keep Q_k as high as possible, we temporarily take $\epsilon'_Q[j,l]$ as the maximum value of $\{\bar{\epsilon}'_Q[j], \bar{\epsilon}'_Q[l]\}$ (Line 4). After curtailing $Q_k[j,l]$ by applying $\epsilon'_Q[j,l]$, we obtain the corresponding lowering factor again (Lines 5–6). For now, the minimum value is taken to maintain Q_k's symmetricity and satisfy (2.25) simultaneously (Line 7). Then, the corresponding stochastic policy is generated based on the resultant Q_k (Lines 8–10).

Note that we set $G_k[j] = 0$ for all time instants and all bins, so $M_k = P_k$.

Algorithm 4 Max Convergence with Flux Limits (Line 4 of Algorithm 2 for P2)

// Initialise P_k

1: $P_k[j,l] = 0$, $\forall l \in \{1, 2, ..., n_b\}$;

// Compute Q_k satisfying the constraint (2.23) only

2: $Q_k[j,l] = \min\left(\frac{c_{(j,l)}}{\mathbf{n}_k[j]\Theta[l]}, \frac{c_{(l,j)}}{\mathbf{n}_k[l]\Theta[j]}\right)$, $\forall \mathcal{B}_l \in \mathcal{N}_k(j) \setminus \{\mathcal{B}_j\}$;

// Lower Q_k to satisfy the constraint (2.25) additionally

3: $\bar{\epsilon}'_Q[j] = \min\left(\frac{\bar{\xi}_k[j]}{\sum_{\forall l \neq j} \Theta[l] Q_k[j,l]}, 1\right)$;

4: $\epsilon'_Q[j,l] = \max\left(\bar{\epsilon}'_Q[j], \bar{\epsilon}'_Q[l]\right)$, $\forall \mathcal{B}_l \in \mathcal{N}_k(j) \setminus \{\mathcal{B}_j\}$;

5: $Q_k[j,l] := \epsilon'_Q[j,l] Q_k[j,l]$, $\forall \mathcal{B}_l \in \mathcal{N}_k(j) \setminus \{\mathcal{B}_j\}$;

6: $\bar{\epsilon}_Q[j] = \min\left(\frac{\bar{\xi}_k[j]}{\sum_{\forall l \neq j} \Theta[l] Q_k[j,l]}, 1\right)$;

7: $\epsilon_Q[j,l] = \min\left(\bar{\epsilon}_Q[j], \bar{\epsilon}_Q[l]\right)$, $\forall \mathcal{B}_l \in \mathcal{N}_k(j) \setminus \{\mathcal{B}_j\}$;

8: $Q_k[j,l] := \epsilon_Q[j,l] Q_k[j,l]$, $\forall \mathcal{B}_l \in \mathcal{N}_k(j) \setminus \{\mathcal{B}_j\}$;

// Compute P_k

9: $P_k[j,l] = \Theta[l] Q_k[j,l]$, $\forall \mathcal{B}_l \in \mathcal{N}_k(j) \setminus \{\mathcal{B}_j\}$;

10: $P_k[j,j] = 1 - \sum_{\forall l \neq j} P_k[j,l]$;

2.4.3 Example III: Local-information-based Quorum Model

This subsection shows that the proposed framework is able to incorporate a quorum model, which is introduced in [11, 12]. In this model, if a bin is overpopulated above a certain level of predefined threshold called *quorum*, the probabilities that agents in the bin move to its neighbour bins are temporarily increased, rather than consistently following given P_k. This feature eventually brings an advantage to the convergence performance of the swarm.

To this end, we set the secondary guidance matrix S_k as follows: $\forall j, l \in \{1, ..., n_b\}$,

$$S_k[j,l] := \begin{cases} 1/|\mathcal{N}_k(j)| & \text{if } \mathbf{A}_k[j,l] = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{2.26}$$

This matrix makes agents in a bin equally disseminated over its neighbour bins. In addition, the secondary feedback gain $G_k[j]$ is defined as

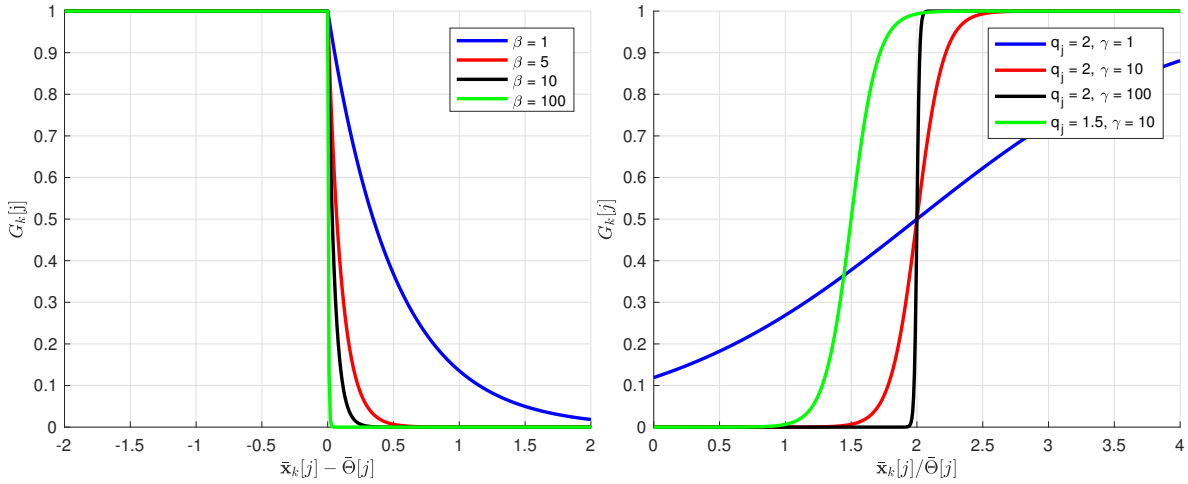
$$G_k[j] := \left(1 + \exp\left(\gamma(q_j - \frac{\bar{\mathbf{x}}_k[j]}{\bar{\Theta}[j]})\right)\right)^{-1}, \tag{2.27}$$

where $\gamma > 0$ is a design parameter, and $q_j > 1$ is the quorum for bin $\mathcal{B}_j$. Figure 2.4(b) shows the gain value varying depending on γ and q_j. As $\bar{\mathbf{x}}_k[j]/\bar{\Theta}[j]$ becomes higher than the quorum, $G_k[j]$ gets close to 1 (i.e., $S_k[j,l]$ becomes more dominant than $P_k[j,l]$). The steepness of $G_k[j]$ around the quorum value is regulated by γ.

The existing quorum models in [11, 12] require each agent to know $\mathbf{x}_k[j]$, which implies that the total number of agents n_a should be tracked in real time. It could be possible that some agents in a swarm unexpectedly become faulted by internal or external effects during a mission, which hinders for other alive agents from keeping track of n_a in a timely manner. On the contrary, this requirement is not the case for the proposed quorum model, and it works by only using local information available from $\mathcal{A}_{\mathcal{N}_k(j)}$.

2.5 Asynchronous Implementation

A synchronous process induces extra time delays and inter-agent communication to make the entire agents, who may have different timescales for obtaining new informa-



(a) P1 **(b)** the Quorum Model

Figure 2.4: The secondary feedback gains $G_k[j]$ depending on the associated design parameters: (a) for P1 (i.e., Eqn. (2.17)); (b) for the quorum model (i.e., Eqn. (2.27))

Algorithm 5 The Quorum Model (Lines 5-6 of Algorithm 2)

1: Compute $S_k[j,l]$ using (2.26), $\forall l$;
2: Compute $G_k[j]$ using (2.27);

tion, remain in sync. Such unnecessary waiting time and communication may cause unfavourable effects on mission performance or even may not be realisable in practice [32].

In the previous sections, it was assumed that a swarm of agents act synchronously at every time instant. Here we show that the proposed framework can accommodate asynchronous behaviours of the agents, assuming that the union of underlying graphs of the corresponding Markov matrices across some time intervals is frequently and infinitely strongly-connected.

Suppose that an algorithm to compute P_k that satisfies (R1)–(R5) in a synchronous environment is given (e.g., Algorithm 3 or 4). We propose an asynchronous implementation as shown in Algorithm 6, which substitutes Line 4 in Algorithm 2. We refer to a set of bins where agents are ready to use their respective local information (e.g., $\mathbf{n}_k[j]$) as $\mathcal{R}_k^+$, and a set of the other bins as $\mathcal{R}_k^-$. For each bin $\mathcal{B}_j$, we denote $\mathcal{N}_k^+(j) := \mathcal{N}_k(j) \cap \mathcal{R}_k^+$

and $\mathcal{N}_k^-(j) := \mathcal{N}_k(j) \cap \mathcal{R}_k^-$. It is assumed that each agent in bin $\mathcal{B}_j \in \mathcal{R}_k^+$ knows the local information of its neighbour bin $\mathcal{B}_l \in \mathcal{N}_k^+(j)$.

In the asynchronous algorithm, each agent in bin $\mathcal{B}_j \in \mathcal{R}_k^+$ follows the existing procedure of generating $P_k[j,l]$ for each neighbour bin $\mathcal{B}_l$ whose local information is also available (Line 2). Then, the probabilities to transition to all the other bins (except $\mathcal{B}_j$) are set to be zero (Line 3). In the meantime, each agent for whom local information is not ready does not deviate but remains at the bin it belongs to. Equivalently, it can be said that $P_k[j,j] = 1$ and $P_k[j,l] = 0$, $\forall l \neq j$ (Line 6).

Hereafter, for the sake of differentiation from the original P_k generated in a synchronous environment, let us refer to the matrix resultant from Algorithm 6 as *asynchronous primary guidance matrix*, denoted by $\tilde{P}_k$. Accordingly, the *asynchronous Markov matrix* can be defined as:

$$\tilde{M}_k := (I - W_k)\tilde{P}_k + W_k S_k.$$

We show that this asynchronous Markov process also converges to the desired swarm distribution.

Lemma 4. *The matrix $\tilde{P}_k$, for every time instant k, satisfies the following properties: (1) row-stochastic; (2) all diagonal elements are positive and all other elements are non-negative; and (3) $\sum_{l=1}^{n_b} \Theta[l]\tilde{P}_k[l,j] = \Theta[j], \forall j$.*

Proof. The matrix $\tilde{P}_k$ is row-stochastic because of Line 4 and 6 in Algorithm 6. Furthermore, given that P_k satisfies (R2), the property (2) is also valid for $\tilde{P}_k$ because

Algorithm 6 Asynchronous Construction of $P_k[j,l]$ (Substitute for Line 4 of Algorithm 2)

1: **if** $\mathcal{B}_j \in \mathcal{R}_k^+$ & **isnonempty**$(\mathcal{N}_k^+(j) \setminus \{\mathcal{B}_j\})$ **then**
2: Compute $P_k[j,l]$ as usual, $\forall \mathcal{B}_l \in \mathcal{N}_k^+(j) \setminus \{\mathcal{B}_j\}$
3: $P_k[j,l] = 0$, $\forall \mathcal{B}_l \in \mathcal{B} \setminus \mathcal{N}_k^+(j)$
4: $P_k[j,j] = 1 - \sum_{\forall l \neq j} P_k[j,l]$
5: **else**
6: $P_k[j,j] = 1$; $P_k[j,l] = 0$, $\forall l \neq j$
7: **end if**

$\tilde{P}_k[j,j] \geq P_k[j,j]$ for $\forall j$.

Let us now turn to the property (3), and firstly consider the case where $\forall \mathcal{B}_j \in \mathcal{R}_k^+$. For any two bins $\mathcal{B}_{j_1}$ and $\mathcal{B}_{j_2}$ ($j_1 \neq j_2$), Algorithm 2 yields that $\tilde{P}_k[j_1,j_2] = P_k[j_1,j_2]$ and $\tilde{P}_k[j_2,j_1] = P_k[j_2,j_1]$ if $\mathcal{B}_{j_1}, \mathcal{B}_{j_2} \in \mathcal{R}_k^+$ and $\mathbf{A}_k[j_1,j_2] = 1$ simultaneously, otherwise $\tilde{P}_k[j_1,j_2] = \tilde{P}_k[j_2,j_1] = 0$. For $\forall \mathcal{B}_j \in \mathcal{R}_k^+$, this fact implies the followings: (i) $\tilde{P}_k[l,j] = P_k[l,j]$ for $\forall \mathcal{B}_l \in \mathcal{N}_k^+(j) \setminus \{\mathcal{B}_j\}$; (ii) $\tilde{P}_k[j,l] = \tilde{P}_k[l,j] = 0$ for $\forall \mathcal{B}_l \in \mathcal{B} \setminus \mathcal{N}_k^+(j)$; and (iii) $\tilde{P}_k[j,j] = P_k[j,j] + \sum_{\forall \mathcal{B}_l \in \mathcal{N}_k^-(j)} P_k[j,l]$. We apply the findings into the following equation:

$$\begin{aligned}\sum_{l=1}^{n_b} \Theta[l]\tilde{P}_k[l,j] = &\sum_{\forall \mathcal{B}_l \in \mathcal{B} \setminus \mathcal{N}_k^+(j)} \Theta[l]\tilde{P}_k[l,j] \\ &+ \sum_{\forall \mathcal{B}_l \in \mathcal{N}_k^+(j) \setminus \{\mathcal{B}_j\}} \Theta[l]\tilde{P}_k[l,j] + \Theta[j]\tilde{P}_k[j,j].\end{aligned} \tag{2.28}$$

The first term of the right hand side becomes zero because of (ii). Due to (i) and the fact that $\Theta[l]P_k[l,j] = \Theta[j]P_k[j,l]$ $\forall l$ (from Requirement 3), the second term becomes $\Theta[j]\sum_{\forall \mathcal{B}_l \in \mathcal{N}_k^+(j)\setminus\{\mathcal{B}_j\}} P_k[j,l]$. The last term becomes $\Theta[j]P_k[j,j] + \Theta[j]\sum_{\forall \mathcal{B}_l \in \mathcal{N}_k^-(j)} P_k[j,l]$ because of (iii). By putting all of them together and adding $\sum_{\forall \mathcal{B}_l \in \mathcal{B}\setminus\mathcal{N}_k(j)} P_k[j,l] = 0$, the right hand side of Equation (2.28) is equivalent to

$$\begin{aligned}&\Theta[j]\left(\sum_{\forall \mathcal{B}_l \in \mathcal{N}_k^+(j)\setminus\{\mathcal{B}_j\}} P_k[j,l] + P_k[j,j] + \sum_{\forall \mathcal{B}_l \in \mathcal{N}_k^-(j)} P_k[j,l] + \sum_{\forall \mathcal{B}_l \in \mathcal{B}\setminus\mathcal{N}_k(j)} P_k[j,l]\right) \\ &= \Theta[j]\sum_{l=1}^{n_b} P_k[j,l] = \Theta[j].\end{aligned}$$

On the other hand, for the case where $\forall \mathcal{B}_j \in \mathcal{R}_k^-$, it follows from Algorithm 6 that $\tilde{P}_k[l,j] = 0, \forall l \neq j$ and $\tilde{P}_k[j,j] = 1$. Thus, $\sum_{l=1}^{n_b} \Theta[l]\tilde{P}_k[l,j] = \Theta[j]$. □

Lemma 5. *If the union of a set of underlying graphs of $\{\tilde{P}_{k_1}, \tilde{P}_{k_1+1}, ..., \tilde{P}_{k_2-1}\}$ is strongly-connected, then the matrix product $\tilde{P}_{k_1,k_2} := \tilde{P}_{k_1}\tilde{P}_{k_1+1}\cdots\tilde{P}_{k_2-1}$ is irreducible.*

Proof. Since the union of a set of underlying graphs of $\{\tilde{P}_{k_1}, \tilde{P}_{k_1+1}, ..., \tilde{P}_{k_2-1}\}$ is strongly-connected, the underlying graph of $\sum_{k=k_1}^{k_2-1} \tilde{P}_k$ is also strongly-connected. Noting that

every $\tilde{P}_k$, $\forall k \in \{k_1, k_1+1, ..., k_2-1\}$ is a nonnegative $n_b \times n_b$ matrix and its diagonal elements are positive (by Lemma 4), it follows from [33, Lemma 2] that $\tilde{P}_{k_1,k_2} \geq \gamma \sum_{k=k_1}^{k_2-1} \tilde{P}_k$, where $\gamma > 0$. This implies that the underlying graph of $\tilde{P}_{k_1,k_2}$ is strongly-connected, and thus the matrix $\tilde{P}_{k_1,k_2}$ is irreducible. □

Theorem 3. *Suppose that there exists an infinite sequence of non-overlapping time intervals* $[k_i, k_{i+1})$, $i = 0, 1, 2, ...$, *such that the union of underlying graphs of* $\{\tilde{P}_{k_i}, \tilde{P}_{k_i+1}, ..., \tilde{P}_{k_{i+1}-1}\}$ *in each interval is strongly-connected. Let the swarm distribution at time instant* $k \geq k_0$, *governed by the corresponding Markov process from an arbitrary state* $\mathbf{x}_{k_0}$, *be* $\mathbf{x}_k = \mathbf{x}_{k_0}\bar{U}_{k_0,k} := \mathbf{x}_{k_0}\tilde{M}_{k_0}\tilde{M}_{k_0+1}\cdots\tilde{M}_{k-1}$. *Then, it holds that* $\lim_{k\to\infty} \mathbf{x}_k = \Theta$ *pointwise for all agents, irrespective of the initial condition.*

Proof. Thanks to Lemma 4 and 5, the matrix product $\tilde{P}_{k_i,k_{i+1}}$ for each time interval $[k_i, k_{i+1})$ satisfies (R1)-(R4). Therefore, one can prove this theorem by similarly following the proof of Theorem 1. □

2.6 Numerical Experiments

2.6.1 Effects of Primary Local-feedback Gain $\bar{\xi}_k[j]$

This section first investigates the sensitivity of the primary feedback gain $\bar{\xi}_k[j]$ using Algorithm 3 (with Equation (2.19)). We show that, depending on the shape of the gain, the performance of the proposed framework changes with respect to convergence rate, fraction of transitioning agents, and residual convergence error.

We consider the scenario having a set of $2,000$ agents and an arena consisting of 10×10 bins, as depicted in Figure 2.2. There are vertical and horizontal paths between adjacent bins. The agents are allowed to move at most 3 paths away within a unit time instant. All the agents start from a bin, which reflects the fact that they are generally deployed from a base station at the beginning of a mission. The desired swarm distribution Θ is uniform-randomly generated at each scenario. The agents are assumed to estimate necessary local information correctly.

The performance of the proposed algorithm will be compared with that of the GICA-based algorithm [10]. To this end, $f_E(E_k[j,l])$ is set to be the same as the corresponding

coefficient in the existing work:

$$f_E(E_k[j,l]) := 1 - \frac{E_k[j,l]}{E_{k,\max} + \epsilon_E}, \tag{2.29}$$

where $E_{k,\max}$ is the maximum element of the travelling expense matrix E_k, and ϵ_E is a design parameter. $E_k[j,l]$ is defined as a linear function based on the distance between bin $\mathcal{B}_j$ and $\mathcal{B}_l$:

$$E_k[j,l] := \epsilon_{E_1} \cdot \Delta s_{(j,l)} + \epsilon_{E_0}, \tag{2.30}$$

where $\Delta s_{(j,l)}$ is the minimum required number of paths from $\mathcal{B}_j$ to $\mathcal{B}_l$; ϵ_{E_1} and ϵ_{E_0} are design parameters. The agents are assumed to follow any shortest route when they transition between two bins. The design parameters are set as follows: $\epsilon_{E_1} = 1$ and $\epsilon_{E_0} = 0.5$ in (2.30); $\epsilon_E = 0.1$ in (2.29); $\epsilon_\xi = 10^{-9}$ in (2.5); $\epsilon_M = 1$ in (2.19); $\beta = 1.8 \times 10^5$ in (2.17); and $\lambda = 10^{-6}$ in (2.8).

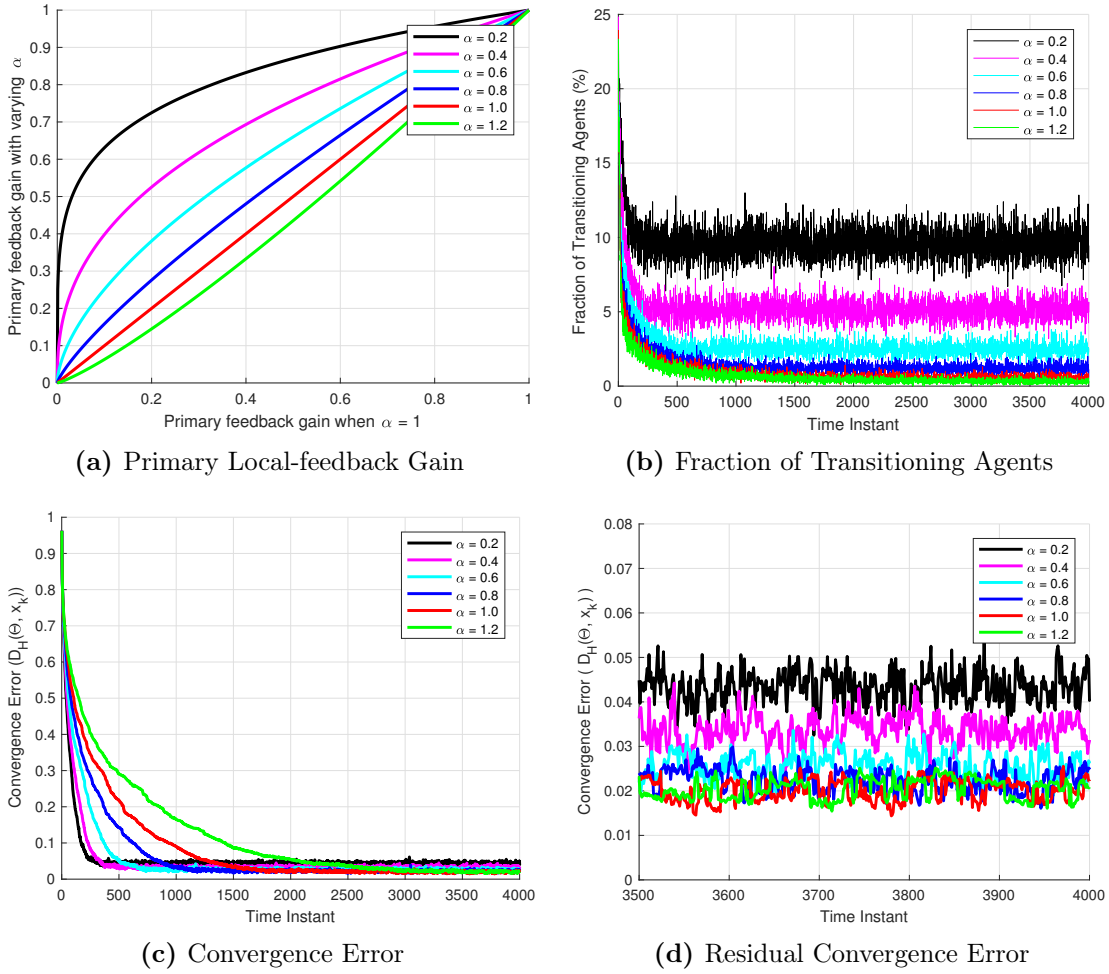
As a performance index for the closeness between $\mathbf{x}_k$ and Θ, we use *Hellinger Distance*, i.e.,

$$D_H(\Theta, \mathbf{x}_k) := \frac{1}{\sqrt{2}} \sqrt{\sum_{j=1}^{n_b} \left(\sqrt{\Theta[j]} - \sqrt{\mathbf{x}_k[j]} \right)^2},$$

which is known as a "concept of measuring similarity between two distributions" [34] and is utilised as a feedback gain in the existing work.

More importantly, to examine the effects of $\bar{\xi}_k[j]$, we set α in (2.5) as $0.2, 0.4, 0.6$, $0.8, 1$ and 1.2.

Figure 2.5 reveals that the convergence rate can be traded off against the fraction of transitioning agents and the residual convergence error. As $\bar{\xi}_k[j]$ becomes more concave (i.e., the value of α decreases), the summation of off-diagonal entries of P_k becomes higher, leading to more transitioning agents but faster convergence rate. At the same time, such unnecessarily higher off-diagonal entries of P_k even at a low value of $|\bar{\Theta}[j] - \bar{\mathbf{x}}_k[j]|$ prevent the agents from properly converging to the desired swarm distribution, resulting in higher residual convergence error.



(a) Primary Local-feedback Gain

(b) Fraction of Transitioning Agents

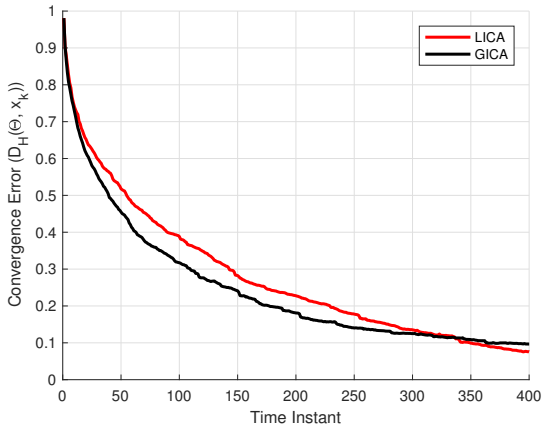
(c) Convergence Error

(d) Residual Convergence Error

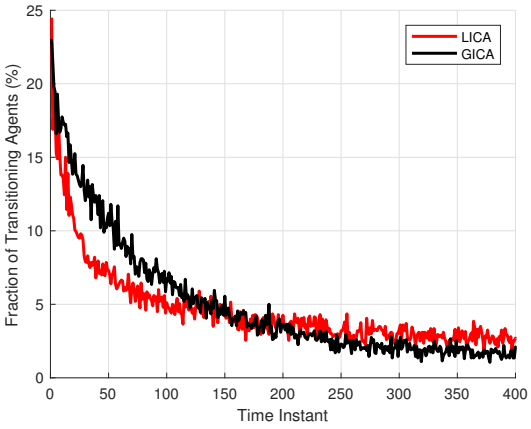
Figure 2.5: Sensitivity analysis depending on the primary local-feedback gain $\bar{\xi}_k[j]$ in Eqn. (2.5) with different setting of α: (a) the value of $\bar{\xi}_k[j]$; (b) the fraction of transitioning agents; (b) the convergence performance; (d) the convergence performance (zoomed-in for time instant between 3500 and 4000)

2.6.2 Comparison with the GICA-based Method

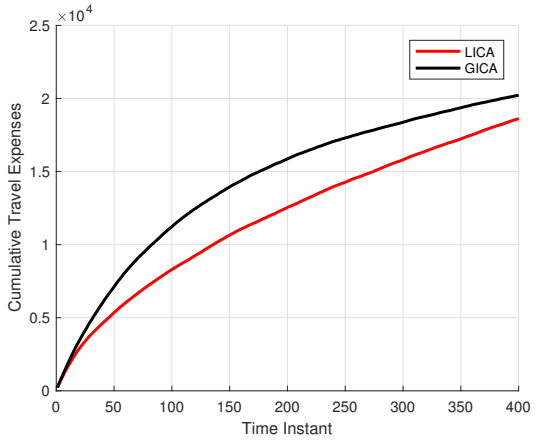
Let us now compare the LICA-based method for (P1) with the GICA-based method. The scenario considered is the same as the one in the previous subsection except for $\alpha = 0.6$. Note that ϵ_Θ in Remark 5 improves convergence rate, but is not discussed in the existing work. For the fair comparison, ϵ_Θ is applied to both methods. We



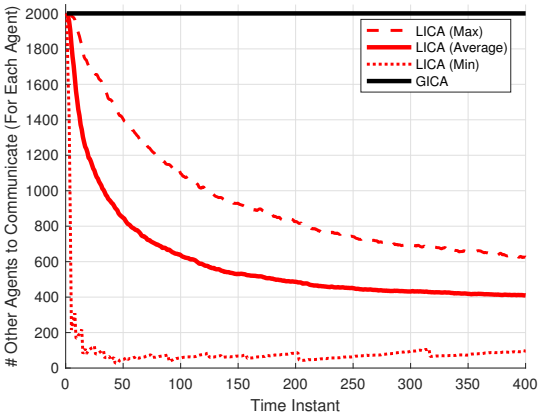
(a) Convergence Error



(b) Fraction of Transitioning Agents



(c) Cumulative Travel Expenses

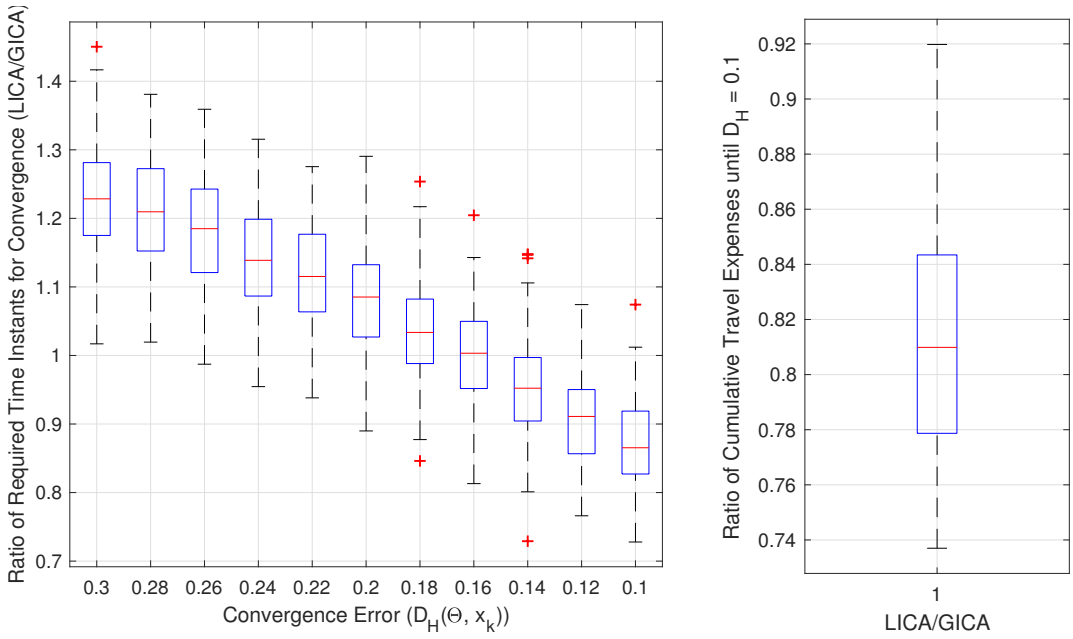


(d) Communication Required

Figure 2.6: Performance comparison between the proposed method (LICA) with the existing method (GICA) [10]: (a) the convergence error from the current swarm status to the desired status; (b) the fraction of agents transitioning between any two bins; (c) the cumulative travel expenses of all the agents from the beginning; (d) the number of other agents whose information are necessary for each agent.

conduct 100 runs of Monte Carlo experiments. Figure 2.6 presents the results of one representative scenario, and Figure 2.7 shows the statistical results of the Monte Carlo experiments.

According to Figure 2.6(a), the convergence rate of the proposed method is comparable to that of the GICA-based method. Specifically, the former is slower at the initial



(a) Converged Time Instants **(b)** Cumulative Expenses

Figure 2.7: Performance comparison (Monte-Carlo experiments) between the proposed method (LICA) and the existing method (GICA) [10]: (a) the required time instants to converge to $D_H(\Theta, \mathbf{x}_k) \in \{0.30, 0.28, ..., 0.12, 0.10\}$ (i.e., convergence rate); (2) the ratio of the cumulative travel expenses by LICA to those by GICA until converging to $D_H(\Theta, \mathbf{x}_k) = 0.1$.

phase, but becomes similar to that of the GICA-based method as reaching $D_H(\Theta, \mathbf{x}_k) = 0.10$. This is confirmed by the statistical results in Figure 2.7(a), which presents the ratio of the required time instants for converging to $D_H(\Theta, \mathbf{x}_k) \in \{0.30, 0.28, ..., 0.12, 0.10\}$ in the LICA-based method to those of the GICA-based method.

Figure 2.6(c) shows that the cumulative travel expenses are smaller in the proposed method than in the existing method. Until achieving $D_H(\Theta, \mathbf{x}_k) = 0.1$, the expenses by the proposed method and those by the compared method are 1.72×10^4 and 1.96×10^4, respectively, and their ratio is 0.878. This is also confirmed by the statistical result in Figure 2.7(b). A possible explanation is that when some of the bins do not meet their desired swarm densities, the entire agents in the GICA-based method would obtain higher feedback gains, leading to unnecessary transitions. On the contrary, this is not the case in the LICA-based method since agents are only affected by their neighbour

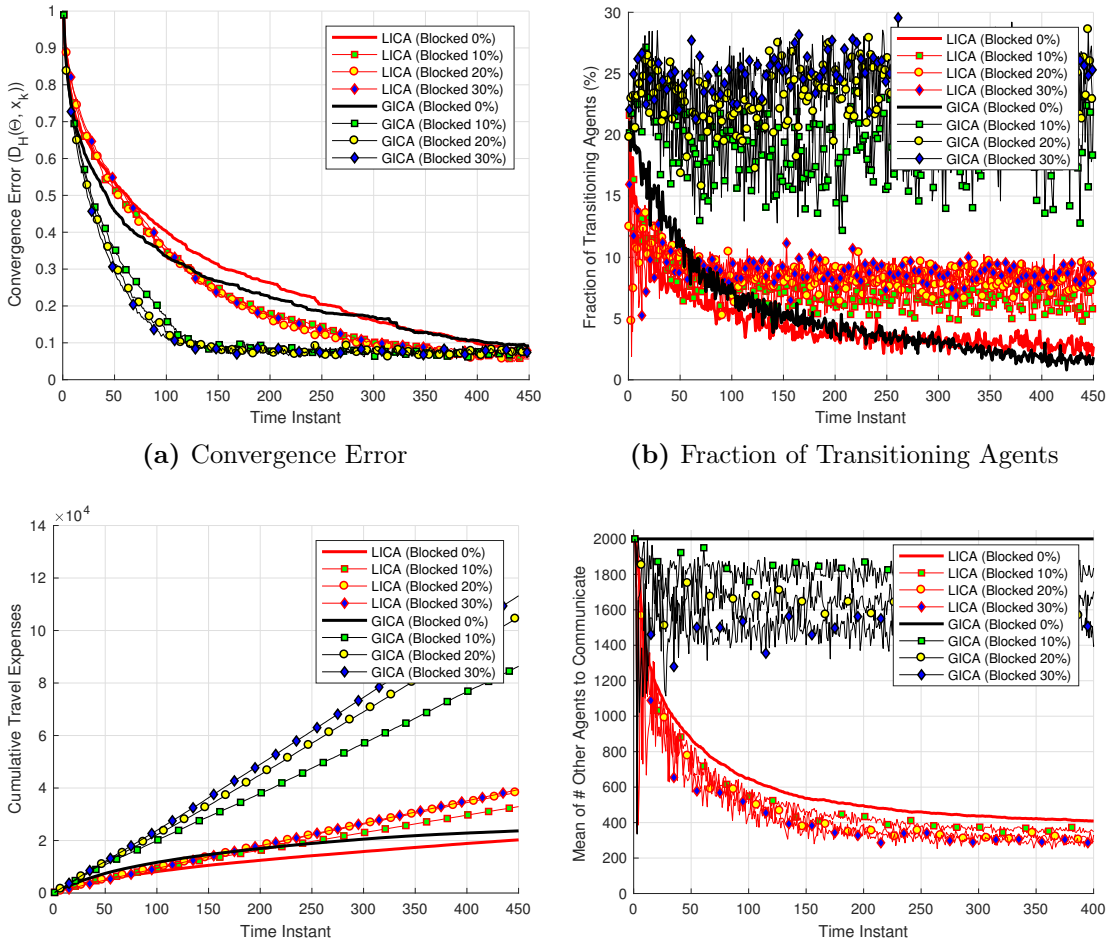
bins.

From the figure, one might argue that as time goes after $D_H(\Theta, \mathbf{x}_k) = 0.1$ is achieved, the cumulative expenses in the proposed method probably exceed those by the GICA-based method. This is mainly because of the current setting of α for the LICA-based method, which induces relatively more transitioning agents at the last phase (as shown in Figure 2.6(b)) as well as a lower level of residual convergence error (as in Figure 2.6(a)) than the GICA-based method. In other words, we might have prevented this possibility by selecting α more carefully. Alternatively, we could also utilise global information once in a while and make the agents forcefully settle down when a certain level of the desired global status is achieved, as proposed in [10].

More importantly, Figure 2.6(d) indicates that agents in the proposed framework require much less information from other agents. This figure shows the number of other agents whose information are necessary for each agent in order to generate its stochastic policies. For the LICA-based framework, the red-dashed line and the red-dotted line represent the maximum case (i.e., the agent who needs the largest amount of information) and the minimum case (i.e., the agent who needs the smallest amount of information) amongst all the agents, respectively. This results show that just 20 % of information are averagely required in the proposed method after the system converges such that $D_H(\Theta, \mathbf{x}_k) < 0.1$, compared with the GICA-based method. This also implies that the LICA-based framework has a shorter timescale for each time instant interval and that its convergence performance in practice would be better. Note that the convergence comparison result in Figure 2.6(a) is presented in respect to time instants of each Markov process.

2.6.3 Robustness in Asynchronous Environments

This subsection investigates the effects of asynchronous environments in the proposed LICA-based method for (P1) and compares them with those in the GICA-based method. We consider a realistic scenario where an asynchronous process is required: agents in some bins cannot communicate or cannot operate temporarily for some reasons (such bins are called *blocked*) and thus other agents in normal bins have to perform their own process without waiting for them. The proportion of blocked bins to the entire bins is



(a) Convergence Error

(b) Fraction of Transitioning Agents

(c) Cumulative Travel Expenses

(d) Communication Required

Figure 2.8: Performance comparison in communication-disconnected situations: (a) the convergence error from the current swarm status to the desired status; (b) the fraction of agents transitioning between any two bins; (c) the cumulative travel expenses of all the agents from the beginning; (d) the average number of other agents whose information are necessary for each agent.

set to be 0%, 10%, 20% and 30%. At each time instant, the corresponding proportion of bins become randomly blocked. Despite no information from the blocked bins, we set that agents in normal bins anyway compute $\bar{\mathbf{x}}_k[j]$ in the proposed method (or $\mathbf{x}_k$ in the GICA-based method). For the proposed method, the asynchronous implementation in Section 2.5 is built with Algorithm 3. The rest of scenario setting are the same as

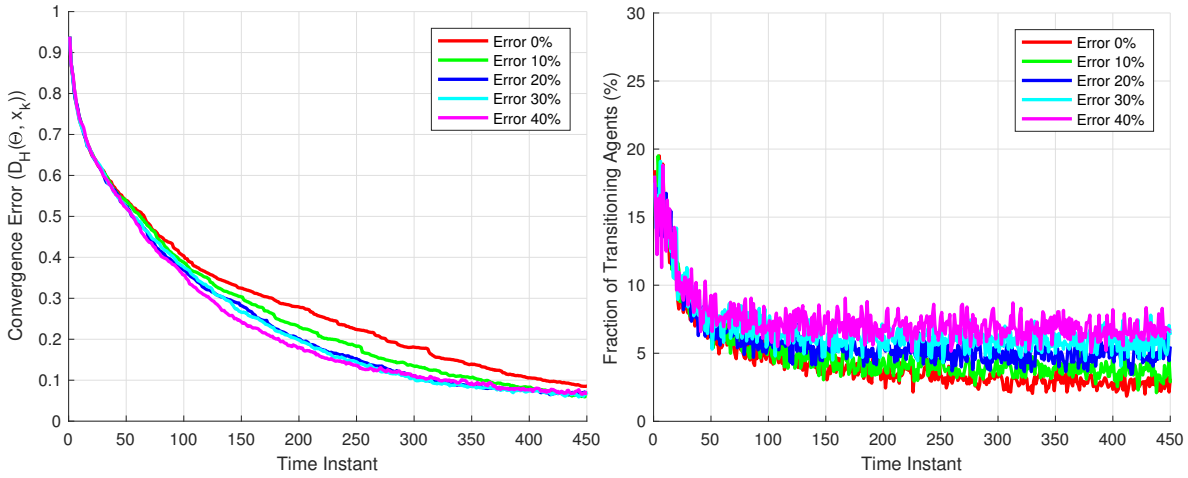
those in Section 2.6.2.

Figure 2.8 illustrates the performance of each method: convergence rate, fraction of transitioning agents, cumulative travel expenses, and the amount of information to communicate. As the proportion of the blocked bins increases, the GICA-based method tends to have faster convergence speed, whereas it loses Desired Feature 2 and thus increases cumulative travelling expenses (as shown in Figure 2.8(a), 2.8(b), and 2.8(c), respectively). On the contrary, the LICA-based method shows a graceful degradation in terms of Desired Feature 2 (as shown in Figure 2.8(b)). A possible explanation for these results could be that higher feedback gains due to the communication disconnection induce faster convergence performance in each method than the normal situation. This effect is dominant for the GICA-based method because it affects the entire agents, who use global information. However, in the LICA-based framework, the communication disconnection only locally influences so that its effectiveness is relatively modest. Figure 2.8(d) shows that the proposed framework still only relies on much less information (i.e., averagely 20% after the system reasonably converges), compared with the existing method.

2.6.4 Effect of Local Information Estimation Error

Let us now examine the performance degradation of the proposed method when there exist estimation errors on local information. In this experiment, agent a_i in bin $\mathcal{B}_j$ is set to locally perceive $\mathbf{n}_k[j]$ as $\mathbf{n}_k^i[j]$, $\forall \mathcal{B}_j \in \mathcal{N}_j(k)$, which is generated from a uniform random distribution $\big[(1-\eta)\cdot\mathbf{n}_k[j], (1+\eta)\cdot\mathbf{n}_k[j]\big]$, where η is the pre-defined error level. Then, $\mathbf{n}_k^i[j]$ is used to compute $\bar{\mathbf{x}}_k[j]$ as in Equation (2.3). Apart from that, we use the same scenario setting in Section 2.6.2, while varying $\eta \in \{0\%, 10\%, 20\%, 30\%, 40\%\}$.

Figure 2.9 shows that despite the estimation errors, the proposed LICA-based method still achieve Desired Features 1 and 2 with graceful performance degradation. The uncertainty induces faster convergence but keeps a higher number of transitioning agents even at the last phase. A possible reason behind this would be similar to that for the results in Section 2.6.3. In practice, such faster convergence behaviour caused by uncertainties provide obvious benefits. The increased transitioning agents at the last phase could be addressed by allowing them to utilise more time to estimate the



(a) Convergence Error **(b)** Fraction of Transitioning Agents

Figure 2.9: Performance degradation of the proposed framework in the existence of estimation error (from 10 % to 40 %) on local information $\mathbf{n}_k[l]$ about neighbour bins: (a) the convergence behaviour; (b) the fraction of transitioning agents.

local information more accurately as the system converges (e.g., by setting variable time instants), considering the fact that the costs of physical transition between bins are in general much more expensive than those for communication.

2.6.5 Demonstration of Example II and III

This subsection demonstrates the LICA-based method for (P2) (i.e., Algorithm 4) and the quorum model (i.e., Algorithm 5). For the former, we consider a scenario where $10,000$ agents and an arena consisting of 10×10 bins are given. The arena is as depicted in Figure 2.2, where the agents are allowed to move only one path away within a unit time instant. For each one-way path, the flux upper limit per time instant is set as 20 agents (i.e., $c_{(j,l)} = 20$, $\forall l \neq j$). All the agents start from a bin, and the desired swarm distribution is uniform-randomly generated.

For the latter, we build the quorum model upon the LICA-based method for (P2). Combining the two may be a good strategy for a user who wants to achieve not only faster convergence rate but also lower unnecessary transitions after equilibrium, which

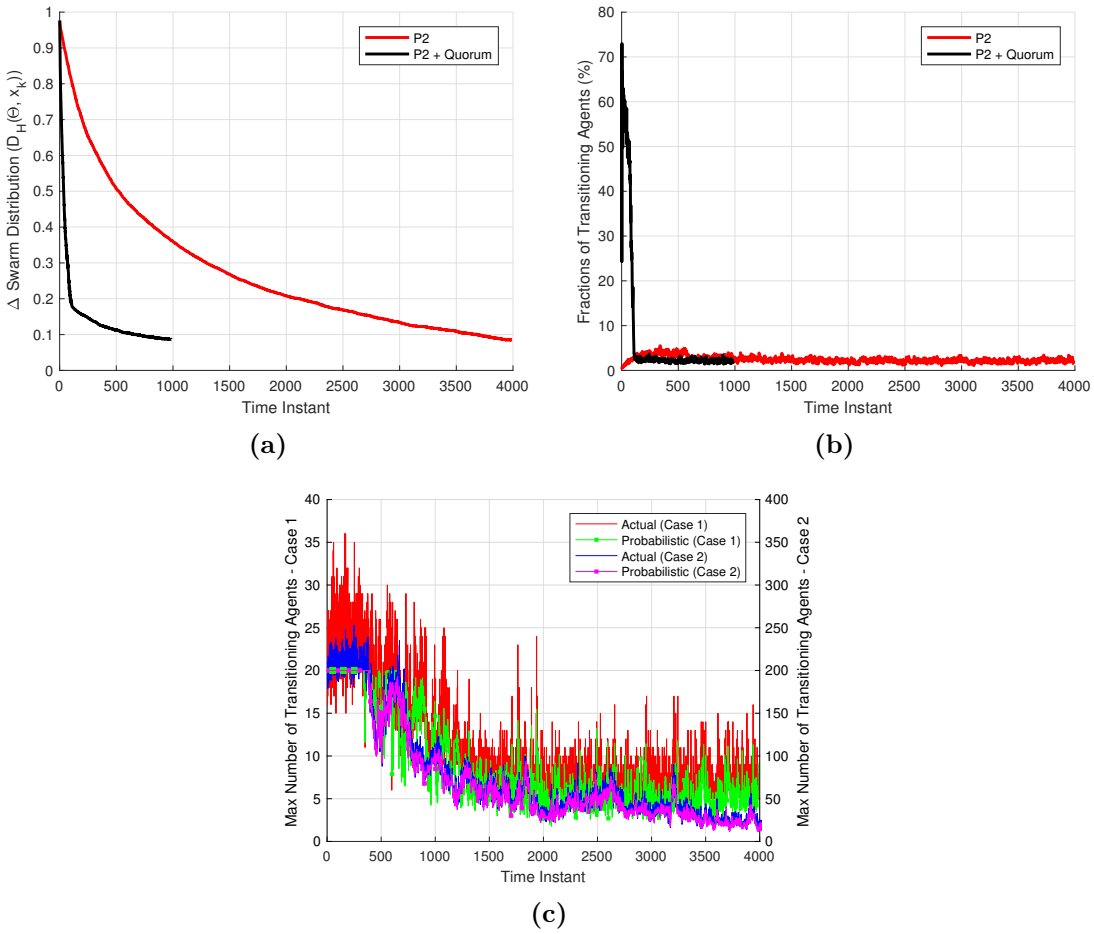


Figure 2.10: Comparison results between the method for (P2) and the quorum-based model: (a) the convergence error from the current swarm status to the desired status; (b) the fraction of agents transitioning between any two bins; (c) The maximum number of transitioning agents via each path in the method for (P2) (Case 1: $|\mathcal{A}| = 10,000$ and $c_{(j,l)} = 20$, $\forall j, \forall l \neq j$; Case 2: $|\mathcal{A}| = 100,000$ and $c_{(j,l)} = 200$, $\forall j, \forall l \neq j$)

are regulated by the flux upper limits. In the same scenario described above, we will demonstrate the combined algorithm that computes S_k and G_k by Algorithm 5 and P_k by Algorithm 4. We set $q_j = 1.3$ and $\gamma = 30$ for (2.27); $\alpha = 1$ and $\epsilon_\xi = 10^{-9}$ for (2.5); and $\lambda = 10^{-6}$ for (2.8).

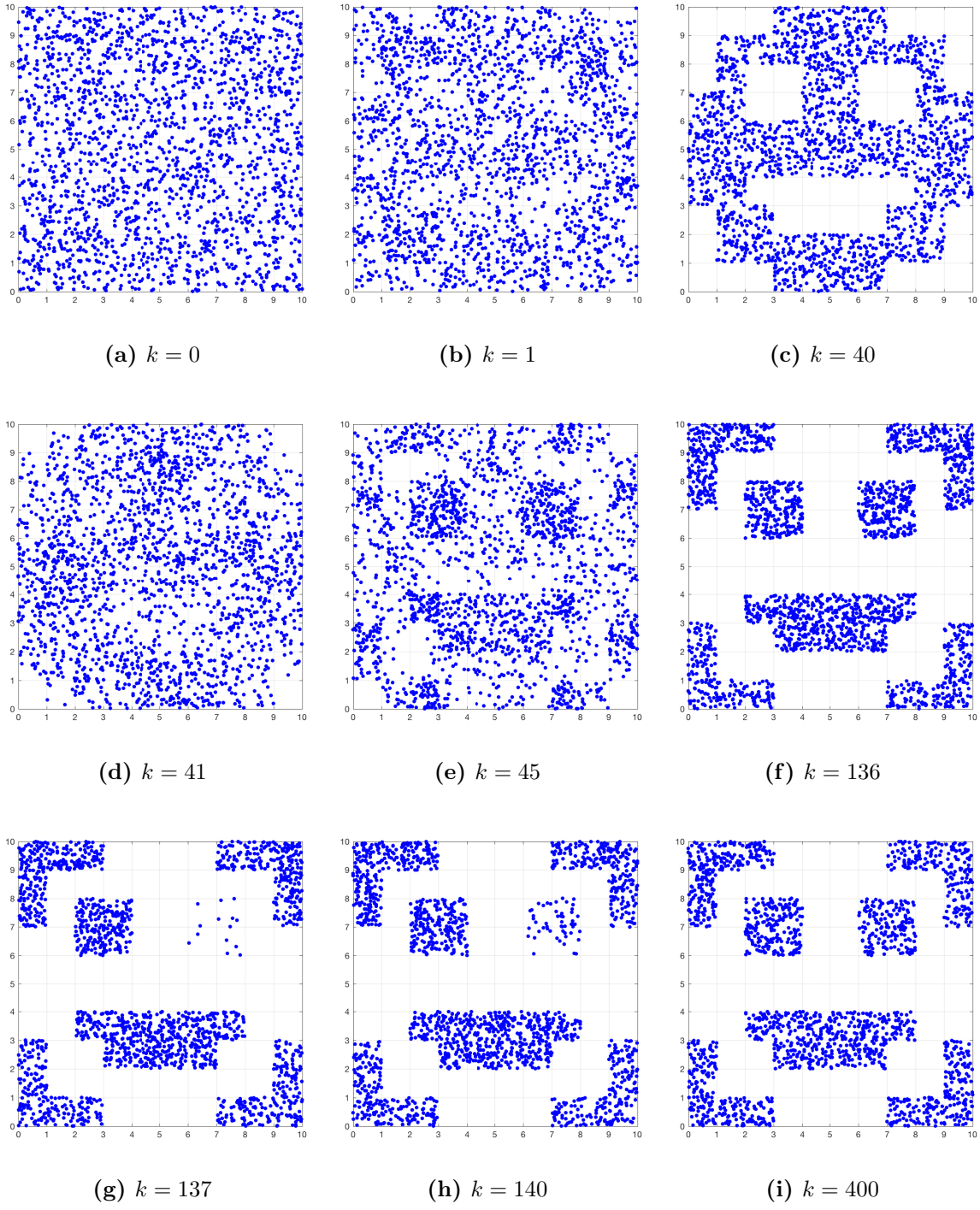
Figure 2.10(a) presents that both approaches make the swarm converge to the desired swarm distribution. It is observed from Figure 2.10(b) that the number of tran-

sitioning agents in the method for (P2) is restricted because of the upper flux bound during the entire process. Meanwhile, the quorum-based method very quickly disseminates the agents, who are initially at one bin, over the other bins, and thus the fraction of transitioning agents is very high at the initial phase. After that, the population fraction drops and remains as low as that by the method for (P2).

Figure 2.10(c) presents the maximum value amongst the number of transitioning agents via each (one-way) path. The red line indicates the actual result by the method for (P2), while the green line indicates the corresponding probabilistic value (i.e., $\max_{\forall j} \max_{\forall l \neq j} \mathbf{n}_k[j] P_k[j,l]$). It is shown that the stochastic policies reflect the given upper bound, meanwhile this bound is often violated in practice due to the finite-number agents' randomness. However, the result in the same scenario with setting $|\mathcal{A}| = 100,000$ and $c_{(j,l)} = 200$, $\forall j, l \neq j$ (denoted by Case 2), depicted by the blue and magenta lines in Figure 2.10(c), suggests that such violation can be mitigated as the number of given agents increases.

2.6.6 Visualised Adaptiveness Test

This section considers a scenario where a swarm of agents are to visually configure certain images as their emergent behaviours. Every pixel of a performance area is regarded as a bin. We have $n_a = 2000$ agents and $n_b = 10 \times 10$ bins, and all the bins are connected as shown in Figure 2.2. The scenario considered is as follows. Initially (i.e., at $k = 0$), all the agents are randomly distributed over the performance area, and they know the desired distribution vector, which is a smile icon. At $k = 41$, the inverted-colour icon is given to them as a new desired distribution vector. Then (i.e., at $k = 137$), it happens that some agents for the right eye of the smiling face are somehow unexpectedly eliminated. This lost of the agents is only informed to their neighbour alive agents, but not to the other far away agents. We use the proposed algorithm for (P1) (i.e., Algorithm 3) and additionally adopt a subroutine whereby agents in zero-desired-density bins randomly move to one of its closest neighbour bins. Due to this subroutine and the constraint $\Theta[j]P_k[j,l] = \Theta[l]P_k[l,j]$ in Requirement 3 (i.e., not allowed to move to a zero-desired-density bin from a positive-desired-density one), all the agents eventually remain in the positive-desired-density bins. The rest of



(a) $k = 0$ **(b)** $k = 1$ **(c)** $k = 40$

(d) $k = 41$ **(e)** $k = 45$ **(f)** $k = 136$

(g) $k = 137$ **(h)** $k = 140$ **(i)** $k = 400$

Figure 2.11: Visualisation results: 2000 agents are deployed into 100 pixels (bins) to configure images.

scenario setting are the same as those in Section 2.6.2. The visualised results are shown in Figure 2.11.

2.7 Conclusion

This chapter proposed a LICA(Local Information Consistency Assumption)-based closed-loop-type Markov-chain framework for probabilistic swarm distribution guidance. For feedback generation, it only requires local information to be known locally. Consequently, agents in the proposed framework exhibit reduced communication transactions, have shorter timescales for using new information, can incorporate an asynchronous decision-making process, and can be deployed into a mission without a priori global knowledge. Even using such insufficient information, it was shown that the agents converge to a desired density distribution, while the framework maintaining scalability, robustness, and long-term system efficiency. The numerical experiments have showed that the proposed framework is more robust in a realistic environment where communication between agents is partially and temporarily disconnected. This chapter has explicitly presented the design requirements for the Markov matrix to hold all these advantages, and has provided specific problem examples of how to implement this framework. As long as the design requirements are satisfied, prospective users can utilise the proposed framework with customising P_k, S_k, $\bar{\xi}_k$, and G_k, depending on their own purposes.

Future work will investigate optimisation of $\bar{\xi}_k[j]$, which can mitigate the trade-off between convergence rate and residual error. In addition, it is expected that the communication cost required for the proposed framework can be reduced by incorporating a vision-based local density estimation [35].

Appendix

A. Proof for Theorem 1

Definition 10 (*Irreducible*)**.** A matrix is *reducible* if and only if its associated digraph is not strongly connected. A matrix that is not reducible is *irreducible*.

Definition 11 (*Primitive*)**.** A *primitive* matrix is a square nonnegative matrix $\mathcal{M}$ such that for every i, j there exists $k > 0$ such that $\mathcal{M}^k[i, j] > 0$.

Definition 12 (*Regular*)**.** A *regular* matrix is a stochastic matrix such that all the entries of some power of the matrix are positive.

Definition 13. [31, pp.92, 149] [10] (*Asymptotic Homogeneity*) "A sequence of stochastic matrices $\mathcal{M}_k \in \mathbb{M}^{n\times n}$, $k \geq k_0$, is said to be *asymptotically homogeneous* (with respect to d) if there exists a row-stochastic vector $d \in \mathbb{P}^n$ such that $\lim_{k\to\infty} d\mathcal{M}_k = d$."

Definition 14. [31, pp.92, 149] [10] (*Strong Ergodicity*) "The forward matrix product $\mathcal{U}_{k_0,k} := \mathcal{M}_{k_0}\mathcal{M}_{k_0+1}\cdots\mathcal{M}_{k-1}$, formed from a sequence of stochastic matrices $\mathcal{M}_r \in \mathbb{P}^{n\times n}$, $r \geq k_0$, is said to be *strongly ergodic* if for each j, l, r we get $\lim_{r\to\infty} \mathcal{U}_{k_0,r}[j, l] = \mathbf{v}[l]$", where $\mathbf{v} \in \mathbb{P}^n$ is a row-stochastic vector. Here, $\mathbf{v}$ is called its *unique limit vector* (i.e., $\lim_{r\to\infty} \mathcal{U}_{k_0,r} = \mathbf{1}^\top\mathbf{v}$).

Lemma 6. *Given the requirements (R1)-(R4) are satisfied, M_k in Equation (2.9) has the following properties:*

1. *row-stochastic;*
2. *irreducible;*
3. *all diagonal elements are positive, and all other elements are non-negative;*
4. *there is a positive lower bound κ such that $0 < \kappa \leq \min^+_{j,l} M_k[j, l]$ (Note that $\min^+$ denotes the minimum of the positive elements);*
5. *asymptotically homogeneous with respect to Θ.*

In addition, $U_{k_0,k}$ in Equation (2.10) has the following properties:

6. *irreducible;*
7. *primitive.*

Proof. This lemma can be proved by similarly following the mathematical development for [10, Theorem 4]. M_k is row-stochastic because $M_k \cdot \mathbf{1}^\top = (I - W_k)P_k \cdot \mathbf{1}^\top +$

$W_k S_k \cdot \mathbf{1}^\top = (I - W_k) \cdot \mathbf{1}^\top + W_k \cdot \mathbf{1}^\top = \mathbf{1}^\top$. P_k is irreducible and $\omega_k[j]$ is always less than 1, thus M_k is also irreducible (i.e., $M_k[j,l] > 0$ if $P_k[j,l] > 0$). The property *3* is true because $\text{diag}(I - W_k) > \mathbf{0}$, $W_k \geq \mathbf{0}$, and P_k is also a non-negative matrix such that its diagonal elements are positive. The property *4* is implied by either the property *2* or *3*. From the definition of W_k (i.e., Equation (4.19)), it follows that $\lim_{k\to\infty} W_k = \mathbf{0}$ because of $\exp(-\lambda k)$, and thereby $\lim_{k\to\infty} M_k = \lim_{k\to\infty} P_k$. Hence, $\lim_{k\to\infty} \Theta M_k = \lim_{k\to\infty} \Theta P_k = \Theta$, and the property *5* is valid.

Let us now turn to $U_{k_0,k}$. If $M_r[j,l] > 0$ for some $r \in \{k_0, ..., k-1\}$ and $j, l \in \{1, ..., n_b\}$, then the corresponding element $U_{k_0,k}[j,l] > 0$ [10, Theorem 4]. Thus, due to the property *2*, $U_{k_0,k}$ is irreducible as well. Besides, it follows from [36, Lemma 8.5.4, p.541] that $U_{k_0,k}$ is primitive: "if a square matrix is irreducible, nonnegative and all its main diagonal entries are positive, then the matrix is primitive". □

Proof for Theorem 1. Theorem 1 can be proved by following similar steps in proving [10, Theorem 4]. The claim in the thorem is true if $\lim_{k\to\infty} \mathbf{x}_k = \mathbf{x}_{k_0} \cdot \lim_{k\to\infty} U_{k_0,k} = \mathbf{x}_{k_0} \cdot \mathbf{1}^\top \Theta = \Theta$. In order for that, the matrix product $U_{k_0,k}$ should (i) be *strongly ergodic* and (ii) have Θ as its unique limit vector, i.e., $\lim_{k\to\infty} U_{k_0,k} = \mathbf{1}^\top \Theta$. We will show that the two conditions are valid under the assumption that (R1)-(R4) are satisfied.

From Lemma 6, we found that (a) $U_{k_0,k}$ is primitive (thus, regular); (b) there is a positive lower bound κ such that $0 < \kappa \leq \min^+_{j,l} M_k[j,l]$, $\forall k$; and (c) M_k is asymptotically homogeneous. Then, it follows from [31, Theorem 4.15, p.150] that $U_{k_0,k}$ is strongly ergodic with respect to a certain vector $\mathbf{v} \in \mathbb{P}^{n_b}$, which fulfils the condition (i).

Let $\mathbf{e}_k \in \mathbb{P}^{n_b}$ be the unique stationary distribution vector corresponding to M_k (i.e., $\mathbf{e}_k M_k = \mathbf{e}_k$). Due to the prior condition (b) and the fact that (d) M_k is irreducible for $\forall k \geq k_0$, it follows from [31, Theorem 4.12, p.149] that the asymptotical homogeneity of M_k with respect to Θ (i.e., $\lim_{k\to\infty} \Theta M_k = \Theta$), given by Lemma 6, is equivalent to both $\lim_{k\to\infty} \mathbf{e}_k = \mathbf{e}$, where $\mathbf{e}$ is a limit vector, and $\Theta = \mathbf{e}$. According to [31, Corollary, p.150], under the prior conditions (b) and (d) and if $U_{k_0,k}$ is strongly ergodic with its unique limit vector $\mathbf{v}$, then $\mathbf{v} = \mathbf{e}$. Hence, it turns out that the unique limit vector of $U_{k_0,k}$ is $\mathbf{v} = \mathbf{e} = \Theta$ (i.e, $\lim_{k\to\infty} U_{k_0,k} = \mathbf{1}^\top \Theta$). Thereby, the condition (ii) is also fulfilled. □

B. Proof for Corollary 3

Proof for Corollary 3. We can prove this by following the proof of [10, Corollary 1]. Suppose that the problem is only subject to (R4) and (2.13), without (R1)-(R3) and (R5). Then, the off-diagonal elements of an optimal matrix should be their corresponding lower bounds in (2.13) if $\mathbf{A}_k[j,l] = 1$. The diagonal elements of the matrix do not affect the objective function due to the fact that $E_k[j,j] = 0, \forall j$. Accordingly, the matrix P_k that holds (2.15) and (2.16) is also an optimal matrix for the simplified problem.

Let us now additionally consider (R1)-(R3) and (R5). Since ϵ_M, $f_\xi(\bar{\xi}_k[j], \bar{\xi}_k[l])$ and $f_E(E_k[j,l])$ are upper-bounded by 1 and $\sum_{\forall l \neq j} \Theta[l] < 1$, $P_k[j,j]$ in (2.16) is always positive for all j, which fulfils (R2). It is also obvious that (R1) is satisfied by Equation (2.16). From Equation (2.15), it holds that $\Theta[j]P_k[j,l] = \Theta[l]P_k[l,j]$, complying with (R3). Since (R1)-(R4) are satisfied, the Markov process is converging to a desired distribution due to Theorem 1. Noting that $f_\xi(\bar{\xi}_k[j], \bar{\xi}_k[l])$ diminishes as $\bar{\xi}_k$ gets close to $\mathbf{0}$ (i.e., $\bar{\mathbf{x}}_k \to \bar{\Theta}$), (R5) is also fulfilled by Equations (2.15) and (2.16). Hence, P_k is the optimal solution for the problem (P1). □

References

[1] E. Sahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 10–20.

[2] B. Acikmese and D. S. Bayard, "A Markov Chain Approach to Probabilistic Swarm Guidance," in *American Control Conference*, Montreal, Canada, 2012, pp. 6300–6307.

[3] B. Acikmese and D. S. Bayard, "Probabilistic Swarm Guidance for Collaborative Autonomous Agents," in *American Control Conference*, Portland, OR, USA, 2014, pp. 477–482.

[4] B. Acikmese and D. S. Bayard, "Markov Chain Approach to Probabilistic Guidance for Swarms of Autonomous Agents," *Asian Journal of Control*, vol. 17, no. 4, pp. 1105–1124, 2015.

[5] I. Chattopadhyay and A. Ray, "Supervised Self-Organization of Homogeneous Swarms Using Ergodic Projections of Markov Chains," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1505–1515, 2009.

[6] N. Demir and B. Acikmese, "Probabilistic Density Control for Swarm of Decentralized ON-OFF Agents with Safety Constraints," in *American Control Conference*, Chicago, IL, USA, 2015, pp. 5238–5244.

[7] R. Luo, N. Chakraborty, and K. Sycara, "Supervisory Control for Cost-Effective Redistribution of Robotic Swarms," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, San Diego, CA, USA, 2014, pp. 596–601.

[8] N. Demir, U. Eren, and B. AçÄśkmeÅ§e, "Decentralized Probabilistic Density Control of Autonomous Swarms with Safety Constraints," *Autonomous Robots*, vol. 39, no. 4, pp. 537–554, 2015.

[9] D. Morgan, G. P. Subramanian, S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic Guidance of Distributed Systems using Sequential Convex

Programming," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA, 2014, pp. 3850–3857.

[10] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1103–1123, 2017.

[11] A. Halasz, M. A. Hsieh, S. Berman, and V. Kumar, "Dynamic Redistribution of a Swarm of Robots among Multiple Sites," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, 2007, pp. 2320–2325.

[12] M. A. Hsieh, A. Halasz, S. Berman, and V. Kumar, "Biologically Inspired Redistribution of a Swarm of Robots among Multiple Sites," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 121–141, 2008.

[13] A. Prorok, M. A. Hsieh, and V. Kumar, "The Impact of Diversity on Optimal Control Policies for Heterogeneous Robot Swarms," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 346–358, 2017.

[14] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Inhomogeneous Markov Chain Approach To Probablistic Swarm Guidance Algorithms," in *5th Int. Conf. on Spacecraft Formation Flying Missions and Technologies (SFFMT)*, Munich, Germany, 2013.

[15] K. Lerman, A. Martinoli, and A. Galstyan, "A Review of Probabilistic Macroscopic Models for Swarm Robotic Systems," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 143–152.

[16] T. W. Mather and M. A. Hsieh, "Macroscopic Modeling of Stochastic Deployment Policies with Time Delays for Robot Ensembles," *The International Journal of Robotics Research*, vol. 30, no. 5, pp. 590–600, 2011.

[17] S. Berman, A. Halasz, M. Ani Hsieh, and V. Kumar, "Navigation-based Optimization of Stochastic Strategies for Allocating a Robot Swarm among Multiple Sites," in *IEEE Conference on Decision and Control*, Cancun, Mexico, 2008, pp. 4376–4381.

[18] S. Berman, A. Halasz, M. A. Hsieh, and V. Kumar, "Optimized Stochastic Policies for Task Allocation in Swarms of Robots," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 927–937, 2009.

[19] L. B. Johnson, H.-L. Choi, and J. P. How, "The Role of Information Assumptions in Decentralized Task Allocation: A Tutorial," *IEEE Control Systems*, vol. 36, no. 4, pp. 45–58, 2016.

[20] I. D. Couzin, J. Krause, R. James, G. D. Ruxton, and N. R. Franks, "Collective Memory and Spatial Sorting in Animal Groups." *Journal of theoretical biology*, vol. 218, no. 1, pp. 1–11, 2002.

[21] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective Leadership and Decision-making in Animal Groups on the Move," *Nature*, vol. 433, no. 7025, pp. 513–516, 2005.

[22] J. Gautrais, C. Jost, and G. Theraulaz, "Key Behavioural Factors in a Self-Organised Fish School Model," *Annales Zoologici Fennici*, vol. 45, no. 5, pp. 415–428, 2008.

[23] D. J. Hoare, I. D. Couzin, J. G. J. Godin, and J. Krause, "Context-dependent Group Size Choice in Fish," *Animal Behaviour*, vol. 67, no. 1, pp. 155–164, 2004.

[24] T. D. Seeley, *The Wisdom of the Hive*. Cambridge, Massachusetts: Havard University Press, 1995.

[25] L. Keller, M. J. B. Krieger, and J.-B. Billeter, "Ant-like Task Allocation and Recruitment in Cooperative Robots," *Nature*, vol. 406, no. 6799, pp. 992–995, 2000.

[26] B. L. Partridge, "The Structure and Function of Fish Schools," *Scientific American*, vol. 246, no. 6, pp. 114–123, 1982.

[27] C. Becco, N. Vandewalle, J. Delcourt, and P. Poncin, "Experimental Evidences of a Structural and Dynamical Transition in Fish School," *Physica A: Statistical Mechanics and its Applications*, vol. 367, pp. 487–493, 2006.

[28] S. Bandyopadhyay and S.-J. Chung, "Distributed Estimation using Bayesian Consensus Filtering," *American Control Conference*, pp. 634–641, 2014.

[29] Y. Bestaoui Sebbane, *Planning and Decision Making for Aerial Robots*, ser. Intelligent Systems, Control and Automation: Science and Engineering. Cham: Springer International Publishing, 2014, vol. 71.

[30] I. C. F. Ipsen and T. M. Selee, "Ergodicity Coefficients Defined by Vector Norms," *SIAM Journal on Matrix Analysis and Applications*, vol. 32, no. 1, pp. 153–200, jan 2011.

[31] E. Seneta, *Non-negative Matrices and Markov Chains*, ser. Springer Series in Statistics. Springer New York, 1981.

[32] L. Johnson, S. Ponda, H.-L. Choi, and J. How, "Asynchronous Decentralized Task Allocation for Dynamic Environments," in *Infotech@Aerospace*, St.Louis, MO, USA, 2011.

[33] A. Jadbabaie, Jie Lin, and A. Morse, "Coordination of Groups of Mobile Autonomous Agents using Nearest Neighbor Rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[34] J. Chung, P. Kannappan, C. Ng, and P. Sahoo, "Measures of Distance between Probability Distributions," *Journal of Mathematical Analysis and Applications*, vol. 138, no. 1, pp. 280–292, 1989.

[35] S. A. M. Saleh, S. A. Suandi, and H. Ibrahim, "Recent Survey on Crowd Density Estimation and Counting for Visual Surveillance," *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 103–114, 2015.

[36] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. Cambridge: Cambridge University Press, 2012.

Chapter 3

Anonymous Hedonic Game for Task Allocation in a Large-Scale Multiple Agent System

3.1 Introduction

Decision-making frameworks for a robotic swarm should be *decentralised* (i.e., the desired collective behaviour can be achieved by individual agents relying on local information), *scalable*, *predictable* (e.g., regarding convergence performance and outcome quality), and *adaptable* to dynamic environments (e.g., unexpected elimination or addition of agents or tasks). The frameworks are also desirable to be *robust* against asynchronous environments because, due to the large cardinality of the system and its decentralisation, it is very challenging for every agent to behave synchronously. Furthermore, it is also preferred to be capable of accommodating different interests of agents (e.g., different swarms operated by different organisations).

To this end, in addition to the previous chapter, we propose another novel decision-making framework based on hedonic games [1–3]. The task allocation problem considered is about, given a set of large number of agents being much larger than the number of tasks, how to partition the agents into subgroups and assign the subgroups to each task. It is assumed that each agent can be assigned to at most one task, whereas

each task may require multiple agents: this case falls into ST-MR (single-task robot and multi-robot task) category [4, 5]. This chapter models the problem considered as a coalition-formation game where self-interest agents are willing to form coalitions to improve their own interests. The objective of this game is to find a *Nash stable* partition, which is a social agreement where all the agents agree with the current task assignment. Despite any possible conflicts between the agents, this chapter shows that if they have *social inhibition*, then a Nash stable partition can always be determined within polynomial times in the proposed framework and all the desirable characteristics mentioned above can be achieved. Furthermore, we analyse the lower bound of the outcome's suboptimality and show that 50% of suboptimality is at least guaranteed for a particular case. Various settings of numerical experiments validate that the proposed framework is scalable, adaptable, and robust even in asynchronous environments.

This chapter is organised as follows. Section 3.2 reviews existing literature on decentralised task allocation schemes for robotic swarms, and introduces a recent finding in hedonic games that inspires this study. Section 3.3 proposes our decision-making framework, named *GRAPE*, and analytically proves the existence of and the polynomial-time convergence to a Nash stable partition. Section 3.4 discusses the proposed framework's algorithmic complexity (i.e., scalability), suboptimality, adaptability, and robustness. Section 3.5 shows that the framework can also address a task allocation problem in which each task may need a certain number of agents for completion. Numerical simulations in Section 3.6 confirm that the proposed framework holds all the desirable characteristics. Finally, concluding remarks are followed in Section 3.7.

3.2 Related Work

3.2.1 Decentralised Coordination of Robotic Swarms

Existing approaches for task allocation problems can be categorised into two branches, depending on how agents eventually reach a converged outcome: *orchestrated* and *(fully) self-organised* approaches [6]. In the former, additional mechanism such as a negotiation or voting model is imposed so that some agents can be worse off if a specific condition is met (e.g., the global utility is better off). Alternatively, in self-organised approaches,

each agent simply makes a decision without negotiating with the other agents. The latter generally induce less resource consumption in communication and computation [7], and hence they are preferable in terms of scalability. On the other hand, the former usually provide a better quality of solutions with respect to the global utility, and a certain level of suboptimality could be guaranteed [8–10]. A comparison result between the two approaches [7] presents that as the available information to agents becomes local, the latter becomes to outperform the former. In the following, we particularly review existing literature on self-organised approaches because, for large-scale multiple agent systems, scalability is at least essential and it is realistic to regard that the agents only know their local information but instead the global information.

Self-organised approaches can be categorised into *top-down approaches* and *bottom-up approaches* according to which level (i.e., between an ensemble or individuals) is mainly focused on. Top-down approaches emphasise developing the macroscopic model for a whole system. For instance, population fractions associated with given tasks are represented as states, and the dynamics of the population fractions is modelled by Markov chains [11–15] or differential equations [16–20]. Given a desired fraction distribution over the tasks, agents can converge to the desired status by following local decision policies (e.g., the associated rows or columns of the current Markov matrix). One advantage of using top-down approaches is predictability of average emergent behaviour with regard to convergence speed and the quality of a stable outcome (i.e., how well the agents converge to the desired fraction distribution). However, such prediction, to the best of our knowledge, can be made mainly numerically. Besides, as top-down generated control policies regulate agents, it may be difficult to accommodate each agent's individual preference. Also, each agent may have to physically move around according to its local policy during the entire decision-making process, and this fact may cause unnecessary time and energy costs for the transitioning.

Bottom-up approaches focus on designing each agent's individual rules (i.e., microscopic models) that eventually lead to a desired emergent behaviour. Possible actions of a single agent can be modelled as a finite state machine [21], and change of behaviours occurs according to a probabilistic threshold model [22]. A threshold model, which determines the decision boundary between two motions, is adjustable based on an agent's past experiences such as the time spent on working a task [6, 23], the suc-

cess/failure rates [21, 24], and direct communication from a central unit [22]. This feature can improve system adaptability, and may have the potential to incorporate each agent's individual interest if required. However, it has been shown in [24–30] that, in order to predict or evaluate an emergent performance of a swarm utilising bottom-up approaches, a macroscopic model for the swarm eventually has to be developed by abstracting the microscopic models.

3.2.2 Hedonic Games

Hedonic games [1–3] model a conflict situation where self-interest agents are willing to form coalitions to improve their own interests. *Nash stability* [3], which is inspired by *Nash equilibria* in other game theories, plays a key role since it yields a social agreement amongst the agents even without having any negotiation between them. Many researchers have investigated conditions under which a Nash stable partition is guaranteed to exist and to be determined [3,31–33]. Amongst them, the works in [32,33] mainly addressed an *anonymous hedonic game*, in which each agent considers the size of a coalition to which it belongs instead of the identities of the members. Recently, Darmann [33] showed that selfish agents who have *social inhibition* (i.e., preference towards a coalition with a fewer number of members) could converge to a Nash stable partition in an anonymous hedonic game. The author also proposed a centralised recursive algorithm that can find a Nash stable partition within $O(n_a^2 \cdot n_t)$ of iterations. Here, n_a is the number of agents and n_t is that of tasks.

3.2.3 Main Contributions

Inspired by the recent breakthrough of [33], we propose a novel decentralised framework that models the task allocation problem considered as an anonymous hedonic game. The proposed framework is a self-organised approach in that agents make decisions according to its local policies (i.e., individual preferences). Unlike top-down or bottom-up approaches reviewed in the previous section, which primarily concentrate on designing agents' decision-making policies either macroscopically or microscopically, our

work instead focuses on investigating and exploiting advantages from socially-inhibitive agents, while simply letting them greedily behave according to their individual preferences. Explicitly, the main contributions of this chapter are as follows:

1. We show that selfish agents with social inhibition, which we refer to as *SPAO* preference (Definition 4), can reach a Nash stable partition within less algorithmic complexity compared with [33]: $O(n_a^2)$ of iterations are required[1].

2. We provide a decentralised algorithm, which is executable under a strongly-connected communication network of agents and even in asynchronous environments. Depending on the network assumed, the algorithmic complexity may be additionally increased by $O(d_G)$, where $d_G < n_a$ is the graph diameter of the network.

3. The suboptimality of a Nash stable partition in term of the global utility is analysed. We firstly present a mathematical formulation to compute the suboptimality lower bound by using the information of a Nash stable partition and agents' individual utilities. Furthermore, we additionally show that 50 % of the suboptimality can be at least guaranteed if the social utility for each coalition is defined as a non-decreasing function with respect to the number of members in the coalition.

4. Our framework can accommodate different agents with different interests as long as their individual preferences hold SPAO.

5. Through various numerical experiments, it is confirmed that the proposed framework is scalable, fast adaptable to environmental changes, and robust even in a realistic situation where some agents are temporarily unable to proceed a decision-making procedure and to communicate with the other agents.

[1]Note that the definition of *iteration* is described in Definition 5. This comparison assumes the fully-connected communication network because the algorithm in [33] is centralised.

Table 3.1: Nomenclature

Symbol	Description
$\mathcal{A}$	a set of n_a agents $\{a_1, a_2, ..., a_{n_a}\}$
$\mathcal{T}^*$	a set of n_t tasks $\{t_1, t_2, ..., t_{n_t}\}$
t_0	the void task (i.e., not to work any task)
$\mathcal{T}$	a set of tasks, $\mathcal{T} = \mathcal{T}^* \cup \{t_0\}$
(t_j, p_j)	a task-coalition pair (i.e. to do task t_j with p_j participants)
$\mathcal{X}$	the set of task-coalition pairs, $\mathcal{X} = \mathcal{X}^* \cup \{t_0\}$, where $\mathcal{X}^* = \mathcal{T}^* \times \{1, 2, ..., n_a\}$
$\mathcal{P}_i$	agent a_i's preference relation over $\mathcal{X}$
$\succ_i$	the strong preference of agent a_i
$\sim_i$	the indifferent preference of agent a_i
$\succeq_i$	the weak preference of agent a_i
Π	a *partition*: a disjoint set that partitions the agent set $\mathcal{A}$, $\Pi = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{n_t}, \mathcal{S}_0\}$
$\mathcal{S}_j$	the task-specific coalition for t_j
$\Pi(i)$	the index of the task to which agent a_i is assigned given Π
d_G	the graph diameter of the agent communication network
$\mathcal{N}_i$	The neighbour agent set of agent a_i, given a communication network

3.3 GRoup Agent Partitioning and placing Event

3.3.1 Problem Formulation

Firstly, we introduce the multi-robot task allocation problem considered in this chapter and underlying assumptions regarding agents and tasks.

Problem 1. Suppose that there exist a set of n_a agents $\mathcal{A} = \{a_1, a_2, ..., a_{n_a}\}$ and a set of tasks $\mathcal{T} = \mathcal{T}^* \cup \{t_0\}$, where $\mathcal{T}^* = \{t_1, t_2, ..., t_{n_t}\}$ is a set of n_t tasks and t_0 is *the void task* (i.e., not to perform any task). Each agent a_i has *the individual utility* $u_i : \mathcal{T} \times |\mathcal{A}| \rightarrow \mathbb{R}$, which is a function of the task to which the agent is assigned and the number of its co-working agents (including itself) $p_j \in \{1, 2, ..., n_a\}$ (called *participants*). The individual utility for t_0 is zero regardless of the participants. Every task requires multiple agents for its completion because each agent is considered to have limited individual capabilities. Thus, an agent can be assigned to at most one task. The objective of this task allocation problem is to find an assignment that maximises *the global utility*, i.e., the sum of individual utilities of the entire agents. The problem described above is defined as follows:

$$\max_{\{x_{ij}\}} \sum_{\forall a_i \in \mathcal{A}} \sum_{\forall t_j \in \mathcal{T}} u_i(t_j, p_j) x_{ij}, \tag{3.1}$$

where for every t_j

$$p_j = \sum_{\forall a_i \in \mathcal{A}} x_{ij},$$

subject to

$$\sum_{\forall t_j \in \mathcal{T}} x_{ij} \leq 1 \quad \forall a_i \in \mathcal{A}, \tag{3.2}$$

$$x_{ij} \in \{0, 1\} \quad \forall a_i \in \mathcal{A}, \forall t_j \in \mathcal{T}. \tag{3.3}$$

Here, x_{ij} is a binary decision variable that represents whether or not task t_j is assigned to agent a_i.

This chapter uses the term *social utility* to indicate the sum of total individual utilities within any agent group.

Assumption 1 (*Homogeneous agents with limited capabilities*)**.** We consider a large-scale multi-robot system that consists of physically homogeneous agents. This can be justified because the realisation of a swarm can be in general achieved through mass production [34]. Hence, each individual utility u_i is concerned with not the combinations of co-working agents, but their cardinality. Despite that, it is worth noting that agents in this chapter may have different preferences with respect to the given tasks (e.g., for an agent, a spatially closer task is more preferred, whereas this may not be the case for another agent). Besides, noting that "mass production favours robots with fewer and cheaper components, resulting in lower cost but also reduced capabilities [35]", we also assume that each agent can be only assigned to perform at most a single task. According to [4], such a robot is called a *single-task* (ST) robot.

Assumption 2 (*Agents' communication*)**.** The communication network of the entire agents is at least *strongly-connected*, i.e., there exists a directed communication path between any two arbitrary agents. Given a network, $\mathcal{N}_i \subseteq \mathcal{A}$ denotes a set of neighbour agents for agent a_i.

Assumption 3 (*Multi-robot-required tasks*)**.** Every task is a *multi-robot* (MR) task, meaning that the task may require multiple robots [4]. For now, we assume that each task can be performed even by a single agent although it may take a long time. However, in Section 3.5, we will also address a particular case in which some of the tasks need at least a certain number of agents for completion.

Assumption 4 (*Agents' pre-known information*)**.** Every agent a_i only knows its own individual utility function u_i with regard to every task t_j, while not being aware of those of other agents. Through communication, however, they can notice which agent currently choses which task, i.e., *partition* (Definition 2). Note that the agents do not necessarily have to know the true partition information at all the time. Each agent may own its locally-known partition information.

3.3.2 Proposed Game-theoretical Approach: GRAPE

Let us transform Problem 1 into an anonymous hedonic game event where every agent selfishly tends to join a coalition according to its preference.

Definition 1 (*GRAPE*)**.** An instance of *GRoup Agent Partitioning and placing Event* (GRAPE) is a tuple $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ that consists of (1) $\mathcal{A} = \{a_1, a_2, ..., a_{n_a}\}$, a set of n_a agents; (2) $\mathcal{T} = \mathcal{T}^* \cup \{t_0\}$, a set of tasks; and (3) $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_{n_a})$, an n_a-tuple of preference relations of the agents. For each agent a_i, $\mathcal{P}_i$ describes its *preference relation* over the set of task-coalition pairs $\mathcal{X} = \mathcal{X}^* \cup \{t_0\}$, where $\mathcal{X}^* = \mathcal{T}^* \times \{1, 2, ..., n_a\}$; a task-coalition pair (t_j, p_j) is interpreted as "to do task t_j with p_j participants". For any task-coalition pairs $x_1, x_2 \in \mathcal{X}$, $x_1 \succ_i x_2$ implies that agent a_i strongly prefers x_1 to x_2, and $x_1 \sim_i x_2$ means that the preference regarding x_1 and x_2 is indifferent. Likewise, $\succeq_i$ indicates the weak preference of agent a_i.

Note that agent a_i's preference relation can be derived from its individual utility $u_i(t_j, p_j)$ in Problem 1. For instance, given that $u_i(t_1, p_1) > u_i(t_2, p_2)$, it can be said that $(t_1, p_1) \succ_i (t_2, p_2)$.

Definition 2 (*Partition*)**.** Given an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ of GRAPE, a *partition* is defined as a set $\Pi = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{n_t}, \mathcal{S}_0\}$ that disjointly partitions the agent set $\mathcal{A}$. Here, $\mathcal{S}_j \subseteq \mathcal{A}$ is the *(task-specific) coalition* for executing task t_j such that $\cup_{j=0}^{n_t} \mathcal{S}_j = \mathcal{A}$ and $\mathcal{S}_j \cap \mathcal{S}_k = \emptyset$ for $\forall j \neq k$. $\mathcal{S}_0$ is the set of agents who choose the void task t_0. Given a partition Π, $\Pi(i)$ indicates the index of the task to which agent a_i is assigned. For example, $\mathcal{S}_{\Pi(i)}$ is the coalition that the agent belongs to.

The objective of GRAPE is to determine a stable partition that all the agents agree with. In this chapter, we seek for a *Nash stable* partition, which is defined as follows:

Definition 3 (*Nash stable*)**.** A partition Π is said to be *Nash stable* if, for every agent $a_i \in \mathcal{A}$, it holds that $(t_{\Pi(i)}, |\mathcal{S}_{\Pi(i)}|) \succeq_i (t_j, |\mathcal{S}_j \cup \{a_i\}|)$, $\forall \mathcal{S}_j \in \Pi$.

In other words, in a Nash stable partition, every agent prefers its current coalition to joining any of the other coalitions. Thus, every agent does not have any conflict within this partition, and no agent will not unilaterally deviate from its current decision.

Remark 1 (*An advantage of Nash stability: low inter-agent communication*)**.** The rationale for the use of Nash stability amongst various stable solution concepts in hedonic games [1, 36–38] is that it can reduce communication burden between agents required to reach a social agreement. In the process of converging to a Nash stable partition,

an agent does not need to get any permission from the other agents when it is willing to deviate. This property may not be the case for the other solution concepts. Therefore, each agent is only required to notify its altered decision without any negotiation with other agents. This fact can reduce inter-agent communication in the proposed approach.

3.3.3 SPAO Preference: Social Inhibition

This section introduces the key condition, called *SPAO*, that enables our proposed approach to provide all the desirable properties described in Section 3.1, and then explains its implications.

Definition 4 (*SPAO*)**.** Given an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ of GRAPE, it is said that the preference relation of agent a_i with respect to task t_j is *SPAO (Single-Peaked-At-One)* if it holds that, for every $(t_j, p_j) \in \mathcal{X}^*$, $(t_j, p_{j_1}) \succeq_i (t_j, p_{j_2})$ for any $p_{j_1}, p_{j_2} \in \{1, ..., n_a\}$ such that $p_{j_1} < p_{j_2}$. Besides, we say that an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ of GRAPE is SPAO if the preference relation of every agent in $\mathcal{A}$ with respect to every task in $\mathcal{T}^*$ is SPAO.

For an example, suppose that $\mathcal{P}_i$ is such that

$$(t_1, 1) \succ_i (t_1, 2) \succeq_i (t_1, 3) \succ_i t_0 \succ_i (t_2, 1) \sim_i (t_1, 4) \succ_i (t_2, 2).$$

This preference relation indicates that agent a_i has $(t_1, 1) \succ_i (t_1, 2) \succeq_i (t_1, 3) \succ_i (t_1, 4)$ for task t_1, and $(t_2, 1) \succ_i (t_2, 2)$ for task t_2. According to Definition 4, the preference relation for each of the tasks holds SPAO. For another example, given that

$$(t_1, 1) \succ_i (t_1, 2) \succeq_i (t_1, 3) \succ_i t_0 \succ_i (t_2, 2) \sim_i (t_1, 4) \succ_i (t_2, 1),$$

the preference relation regarding task t_1 holds SPAO, whereas this is not the case for task t_2 because of $(t_2, 2) \succ_i (t_2, 1)$. Apart from that, in any of the two examples above, the agent prefers not to work instead of choosing $(t_2, 1)$, $(t_2, 2)$, or $(t_1, 4)$.

This chapter only considers the case in which every agent has SPAO preference relations regarding all the given tasks. Such agents prefer to execute a task with smaller number of collaborators, namely, they have *social inhibition*.

Remark 2 (*Implications of SPAO*)**.** SPAO implies that an agent's individual utility should be a monotonically decreasing function with respect to the size of a task-specific coalition. In practice, SPAO can often emerge. For instance, experimental and simulation results in [39, Figures 3 and 4] show that the total work capacity resulted from cooperation of multiple robots does not proportionally increase due to interferences of the robots. In such a *non-superadditive* environment [40], an agent's individual utility, which represents its work efficiency, monotonically drops as the number of collaborators enlarges even though the social utility is increased. For another example, SPAO also arises when individual utilities are related with shared-resources. As more agents use the same resource simultaneously, their individual productivities become diminished (e.g., traffic affects travel times [41] [42, Example 3]): this situation can be interpreted by *the law of diminishing returns* in economics. As the authors in [40] pointed out, a non-superadditive case is more realistic than a superadditive case: agents in a superadditive environment always attempt to form the grand coalition whereas those in a non-superadditive case are willing to reduce unnecessary costs.

Remark 3 (*Cooperation of selfish agents with different interests*)**.** Selfish agents with different interests can be accommodated in the proposed framework as long as their individual preferences hold SPAO. This implies that the framework may be utilised for a combination of swarm systems from different organisations under the condition that the multiple systems satisfy SPAO.

3.3.4 Existence of and Convergence to a Nash Stable Partition

To begin with, let us provide a counter example showing that a Nash stable partition may not be determined if SPAO does not hold. Given an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$, where $\mathcal{A} = \{a_1, a_2\}$, $\mathcal{T} = \{t_1, t_2\}$, and $\mathcal{P}$ are as follows:

$$\text{For } a_1, \quad \{t_1, 1\} \succ_1 \{t_2, 1\} \succ_1 \{t_1, 2\} \succ_1 \{t_2, 2\} \succ_1 t_0,$$

$$\text{For } a_2, \quad \{t_2, 2\} \succ_2 \{t_1, 1\} \succ_2 \{t_1, 2\} \succ_2 \{t_2, 1\} \succ_2 t_0.$$

Here, agent a_2, who does not hold SPAO with regard to any of the given tasks, tends to be together with agent a_1. Whereas, agent a_1 always try to avoid agent a_2 in any case. Consequently, a Nash stable partition does not occur in this example.

For now, let us prove that if an instance of GRAPE holds SPAO, there always exists a Nash stable partition and it can be found within polynomial time.

Definition 5 (*Iteration*)**.** This chapter uses the term *iteration* to represent an iterative stage in which an arbitrary agent compares the set of selectable task-coalition pairs given an existing partition, and then determines whether or not to join another coalition.

Assumption 5 (*Mutual exclusion algorithm*)**.** We assume that, at each iteration, a single agent exclusively makes a decision and updates the current partition Π if necessary. This chapter refers to this agent as the *deciding agent* at the iteration. Based on the resultant partition, another deciding agent also performs the same process at the next iteration, and this process continues until every agent does not deviate from a specific partition, which is, in fact, a Nash stable partition. To implement this algorithmic process in practice, the agents need a *mutual exclusion* (or called *mutex*) algorithm to choose the deciding agent at each iteration. In this section, for simplicity of description, we assume that all the agents are fully-connected, by which they somehow select and know the deciding agent. However, in Section 3.3.5, we will present a distributed mutex algorithm that enables the proposed approach to be executed under a strongly-connected communication network even in an asynchronous manner.

Lemma 1. *Given an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ of GRAPE that is SPAO, suppose that a new agent $a_r \notin \mathcal{A}$ holding SPAO preference relations with regard to every task in $\mathcal{T}$ joins $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ in which a Nash stable partition is already established. Then, the new instance $(\tilde{\mathcal{A}}, \mathcal{T}, \mathcal{P})$, where $\tilde{\mathcal{A}} = \mathcal{A} \cup \{a_r\}$, also (1) satisfies SPAO; (2) contains a Nash stable partition; and (3) the maximum number iterations required to re-converge to a Nash stable partition is $|\tilde{\mathcal{A}}|$.*

Proof. Given a partition Π, for agent a_i, *the number of additional co-workers tolerable in its coaltiion* is defined as:

$$\Delta_{\Pi(i)} := \min_{\forall \mathcal{S}_j \in \Pi \setminus \{\mathcal{S}_{\Pi(i)}\}} \max_{\Delta \in \mathbb{Z}} \left\{ \Delta \mid (t_{\Pi(i)}, |\mathcal{S}_{\Pi(i)}| + \Delta) \succeq_i (t_j, |\mathcal{S}_j \cup \{a_i\}|) \right\}. \tag{3.4}$$

Due to the SPAO preference relations, this value satisfies the following characteristics: (a) if Π is Nash stable, for every agent a_i, it holds that $\Delta_{\Pi(i)} \geq 0$; (b) $\Delta_{\Pi(i)} < 0$ implies that agent a_i is willing to deviate to another coalition at a next iteration; and (c) for

the agent a_i who deviated at the last iteration and updated the partition as Π', it holds that $\Delta_{\Pi'(i)} \geq 0$.

From Definition 4, it follows that the new instance $(\tilde{\mathcal{A}}, \mathcal{T}, \mathcal{P})$ still holds SPAO. Let Π_0 denote a Nash stable partition in the original instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$. When a new agent $a_r \notin \mathcal{A}$ decides to execute one of the tasks in $\mathcal{T}$ and creates a new partition Π_1, it holds that $\Delta_{\Pi_1(r)} \geq 0$, as shown in (c). If there is no existing agent $a_q \in \mathcal{A}$ whose $\Delta_{\Pi_1(q)} < 0$, then the new partition Π_1 is Nash stable.

Suppose that there exists at least an agent a_q whose $\Delta_{\Pi_1(q)} < 0$. Then, the agent must be one of the existing members in the task-specific coalition that agent a_r selected in the last iteration. As agent a_q moves to another coalition and creates a new partition Π_2, the previously-deviated agent a_r holds $\Delta_{\Pi_2(r)} \geq 1$. In other words, an agent who deviates to a coalition and expels one of the existing agents will not deviate again even if another agent joins the coalition in a next iteration. This implies that at most $|\tilde{\mathcal{A}}|$ of iterations are required to hold $\Delta_{\tilde{\Pi}(i)} \geq 0$ for every agent $a_i \in \tilde{\mathcal{A}}$, where the partition $\tilde{\Pi}$ is Nash stable. □

Lemma 1 is essential not only for the existence of and convergence to a Nash stable partition but also for fast adaptability to dynamic environments.

Theorem 1 (*Existence*)**.** *If $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ is an instance of GRAPE holding SPAO, then a Nash stable partition always exists.*

Proof. This theorem will be proved by induction. Let $M(n)$ be the following mathematical statement: for $|\mathcal{A}| = n$, if an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ of GRAPE is SPAO, then there exists a Nash stable partition.

Base case : When $n = 1$, there is only one agent in an instance. This agent is allowed to participate in its most preferred coalition, and the resultant partition is Nash stable. Therefore, $M(1)$ is true.

Induction hypothesis : Assume that $M(k)$ is true for a positive integer k such that $|\mathcal{A}| = k$.

Induction step : Suppose that a new agent $a_i \notin \mathcal{A}$ whose preference relation regarding every task in $\mathcal{T}$ is SPAO joins the instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$. This induces a new instance $(\tilde{\mathcal{A}}, \mathcal{T}, \mathcal{P})$ where $\tilde{\mathcal{A}} = \mathcal{A} \cup \{a_i\}$ and $|\tilde{\mathcal{A}}| = k + 1$. From Lemma 1, it follows that the

new instance also satisfies SPAO and has a Nash stable partition $\tilde{\Pi}$. Consequently, $M(k+1)$ is true. By mathematical induction, $M(n)$ is true for all positive integers $n \geq 1$. □

Theorem 2 (*Convergence*)**.** *If $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ is an instance of GRAPE holding SPAO, then the number of iterations required to determine a Nash stable partition is at most $|\mathcal{A}| \cdot (|\mathcal{A}| + 1)/2$.*

Proof. Suppose that, given a Nash stable partition in an instance where there exists only one agent, we add another arbitrary agent and find a Nash stable partition for this new instance, and repeat the procedure until all the agents in $\mathcal{A}$ are included. From Lemma 1, if a new agent joins an instance in which the current partition is Nash stable, then the maximum number of iterations required to find a new Nash stable partition is the number of the existing agents plus one. Therefore, it is trivial that the maximum number of iterations to find a Nash stable partition of an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ is given as

$$\sum_{k=1}^{|\mathcal{A}|} k = |\mathcal{A}| \cdot (|\mathcal{A}| + 1)/2. \tag{3.5}$$

Note that this polynomial-time convergence still holds even if the agents are initialised to a random partition. Suppose that we have the following setting: the entire agents $\mathcal{A}$ are firstly not movable from the existing partition, except a set of free agents $\mathcal{A}' \subseteq \mathcal{A}$; whenever the agents $\mathcal{A}'$ find a Nash stable partition Π', one arbitrary agent in $a_r \in \mathcal{A} \setminus \mathcal{A}'$ additionally becomes liberated and deviates from the current coalition $\mathcal{S}_{\Pi'(r)}$ to another coalition in Π'. In this setting, from the viewpoint of the agents in $\mathcal{A}' \setminus \mathcal{S}_{\Pi'(r)}$, the newly liberated agent is considered as *the new agent* in Lemma 1. Accordingly, we can still utilise the lemma for the agents in $\mathcal{A}' \setminus S_{\Pi'(r)} \cup \{a_r\}$. The agents also can find a Nash stable partition if one of them moves to $S_{\Pi'(r)}$ during the process, because, due to a_r, it became $\Delta_{\Pi'(i)} \geq 1$ for every agent $a_i \in S_{\Pi'(r)} \setminus \{a_r\}$. In a nutshell, the agents $\mathcal{A}' \cup \{a_r\}$ can converge to a Nash stable partition within $|\mathcal{A}' \cup \{a_r\}|$, which is equivalent to Lemma 1. Hence, Theorem 1 and this theorem are also valid for the case when the initial partition of the agents are randomly given. □

3.3.5 Decentralised Algorithm

In the previous section, it was assumed that only one agent is somehow chosen to make a decision at each iteration under the fully-connected network. On the contrary, in this section, we propose a decentralised algorithm, as shown in Algorithm 1, in which every agent does decision making based its local information and affects its neighbours simultaneously under a strongly-connected network. Despite that, we show that Theorems 1 and 2 still hold thanks to our proposed distributed mutex subroutine shown in Algorithm 2. The details of the decentralised main algorithm are as follows.

Each agent a_i has local variables such as Π^i, $\mathsf{satisfied}^i$, r^i, and s^i (Line 1–2). Here, Π^i is the agent's locally-known partition; $\mathsf{satisfied}^i \in \{0,1\}^{n_a \times 1}$ is a binary-variable vector that indicates whether or not each agent is satisfied with Π^i; $r^i \in \mathbb{Z}^+$ is an integer variable to represent how many times Π^i has evolved (i.e., the number of iterations happened for updating Π^i until that moment); and $s^i \in [0,1]$ is a uniform-random variable that is generated whenever Π^i is newly updated (i.e., a random time stamp).

Given Π^i, agent a_i examines which coalition is the most preferred amongst others, assuming that other agents remain at the existing coalitions (Line 5). Then, the agent joins the newly found coalition if it is strongly preferred than the existing coalition (Line 6). In this case, the agent updates Π^i to reflect its new decision, increases r^i, and generates a new random time stamp s^i (Lines 7–10). In any case, the agent ascertained that the currently-selected coalition is the most preferred, so the agent becomes satisfied with Π^i (Line 12). Then, agent a_i generates and sends a message $\mathsf{msg}^i = \{r^i, s^i, \Pi^i, \mathsf{satisfied}^i\}$ to its neighbour agents, and vice versa (Line 14).

As such, every agent locally updates its locally-known partition simultaneously. Thus, amongst the newly-updated partitions, only one should be regarded as if it were the partition updated by a *deciding agent* (defined in Assumption 5) at the previous iteration. We refer to this partition as *the valid partition* at the iteration. The distributed mutex subroutine in Algorithm 2 enables the agents to recognise the valid partition even under a strongly-connected network and in asynchronous environments. Before executing this subroutine, each agent a_i collects all the messages received from its neighbour agents $\forall a_k \in \mathcal{N}_i$ (including msg^i) as $\mathcal{M}^i_{rcv} = \{\mathsf{msg}^i, \forall \mathsf{msg}^k\}$. Using this message set, the agent examines whether or not its own partition Π^i is valid (Lines

Algorithm 1 Decision-making algorithm for each agent a_i

// Initialisation

$\mathsf{satisfied}^i \leftarrow \mathbf{0}_{n_a \times 1}$; $r^i \leftarrow 0$; $s^i \leftarrow 0$

$\Pi^i :=$ the initial partition (e.g., $\{\mathcal{S}_0 = \mathcal{A}, \mathcal{S}_j = \emptyset \ \forall t_j \in \mathcal{T}\}$)

// Decision-making process begins

while $\mathsf{satisfied}^i \neq \mathbf{1}_{n_a \times 1}$ **do**

// Make a new decision if necessary

  if $\mathsf{satisfied}^i[i] = 0$ **then**

    $(t_{j*}, |\mathcal{S}_{j*}|) \leftarrow \arg\max_{\forall \mathcal{S}_j \in \Pi^i} \{u_i(t_j, |\mathcal{S}_j \cup \{a_i\}|)\}$

    if $(t_{j*}, |\mathcal{S}_{j*}|) \succ_i (t_{\Pi^i(i)}, |\mathcal{S}_{\Pi^i(i)}|)$ **then**

      Join $\mathcal{S}_{j*}$ and update Π^i

      $r^i \leftarrow r^i + 1$

      $s^i \in \text{unif}[0, 1]$

      $\mathsf{satisfied}^i \leftarrow \mathbf{0}_{n_a \times 1}$

    end if

    $\mathsf{satisfied}^i[i] = 1$

  end if

// Broadcast the local information to neighbour agents

  Broadcast $\mathsf{msg}^i = \{r^i, s^i, \Pi^i, \mathsf{satisfied}^i\}$ and receive msg^k from its neighbours $\forall a_k \in \mathcal{N}_i$

// Select the valid partition from all the received messages

  Collect all the messages $\mathcal{M}^i_{rcv} = \{\mathsf{msg}^i, \forall \mathsf{msg}^k\}$

  $\{r^i, s^i, \Pi^i, \mathsf{satisfied}^i\} :=$ D-MUTEX$(\mathcal{M}^i_{rcv})$

end while

2–9 in Algorithm 2). If there exists any other partition Π^k such that $r^k > r^i$, then the agent considers Π^k more valid than Π^i and keeps the corresponding information r^k, s^k, and $\mathsf{satisfied}^k$. This also happens if $r^k = r^i$ and $s^k > s^i$, i.e., Π^k and Π^i have evolved the same amount of times, but the former has a higher time stamp, which implies that they are not the same. After completing this subroutine, depending on $\mathsf{satisfied}^i[i]$, each agent proceeds the decision-making process again (i.e., Line 4–13 in Algorithm 1) and/or just broadcasts the existing locally-known partition to its neighbour agents (Line 14 in Algorithm 1).

Algorithm 2 Distributed Mutex Subroutine

1: **function** D-MUTEX($\mathcal{M}^i_{rcv}$)
2: **for** each message $\mathsf{msg}^k \in \mathcal{M}^i_{rcv}$ **do**
3: **if** $(r^k > r^i)$ or $(r^k = r^i \ \&\ s^k > s^i)$ **then**
4: $r^i \leftarrow r^k$
5: $s^i \leftarrow s^k$
6: $\Pi^i \leftarrow \Pi^k$
7: $\mathsf{satisfied}^i \leftarrow \mathsf{satisfied}^k$
8: **end if**
9: **end for**
10: **return** $\{r^i, s^i, \Pi^i, \mathsf{satisfied}^i\}$
11: **end function**

In a nutshell, the distributed mutex algorithm makes sure that there is only one valid partition that dominates (or will finally dominate depending on the communication network) any other partitions. In other words, multiple partitions locally evolve, but one of them only eventually survives as long as a strongly-connected agents communication network is given. From each partition's viewpoint, it can be regarded as being evolved by a random sequence of the agents under the fully-connected network. Thus, the partition becomes Nash stable within the polynomial time as shown in Theorem 2. In an extreme case, we may encounter multiple Nash stable partitions at the very last. Nevertheless, according to the mutex algorithm, one of them can be distributedly selected by the agents. All the features imply that agents using Algorithm 1 can find a Nash stable partition in a decentralised manner and Theorems 1 and 2 still hold.

3.4 Analysis

3.4.1 Algorithmic Complexity (Scalability)

Firstly, let us discuss about the running time for the proposed framework to find a Nash stable partition. This chapter refers to a unit time required for each agent to proceed the main loop of Algorithm 1 (Line 4-16) as *time step*. Depending on the communication network considered, especially if it is not fully-connected, it may be possible that some of the given agents have to execute this loop to just propagate their locally-known partition information without affecting r_i as Line 8. Because this process also spends a unit time step, we call it as *dummy iteration* to distinguish from a *(normal) iteration*, which increases r_i.

Notice that such dummy iterations happen sequentially at most d_G times before a normal iteration occurs, where d_G is the graph diameter of the communication network. Hence, thanks to Theorem 2, the total required time steps until finding a Nash stable partition is $O(d_G n_a^2)$. For the fully-connected network case, it becomes $O(n_a^2)$ because of $d_G = 1$. Note that this algorithmic complexity is less than that of the centralised algorithm, i.e., $O(n_a^2 \cdot n_t)$, in [33].

Every agent at each iteration investigates $n_t + 1$ of selectable task-coalition pairs including t_0 given a locally-known valid partition (as shown in Line 5 in Algorithm 1). Therefore, the computational overhead for the agent is $O(n_t)$ per any iteration. With consideration of the total required time steps, the running time of the proposed approach for an agent can be bounded by $O(d_G n_t n_a^2)$. Note that the running time in practice can be much less than the bound since Theorem 2 was conservatively analysed, as described in the following remark.

Remark 4 (*The number of required iterations in practice*)**.** Algorithm 1 allows the entire agents in $\mathcal{A}$ to be involved in the decision-making process, whereas, in the proof for Theorem 2, a new agent can be involved after a Nash stable partition of existing agents is found. Since agents using Algorithm 1 do not need to find every Nash stable partition for each subset of the agents, unnecessary iterations can be reduced. Hence, the number of required iterations in practice may become less than that shown in

Theorem 2, which is also supported by the experimental results in Section 3.6.2.

Let us now discuss about communication overhead for each agent per iteration. Given a network, agent a_i should communicate with $|\mathcal{N}_i|$ of its neighbors, and the size of each message grows with regard to n_a. Hence, the communication overhead of the agent is $O(|\mathcal{N}_i| \cdot n_a)$. It could be quadratic if $|\mathcal{N}_i|$ increases in proportional to n_a. However, this would not happen in practice, for example, with consideration of a spatial distribution of agents, but instead $|\mathcal{N}_i|$ would be saturated.

Remark 5 (*Communication overhead vs. Running time*)**.** To reduce the communication overhead, we may impose *the maximum number of transactions per iteration*, denoted by n_c, on each agent. Even so, Theorems 1 and 2 are still valid as long as the union of underlying graphs of the communication networks over time intervals becomes connected. However, in return, the number of dummy iterations may increase, so does the framework's running time. In an extreme case such that $n_c = 1$ (i.e., unicast mode), dummy iterations may happen in a row at most n_a times. Thus, the total required time steps until finding a Nash stable partition could be $O(n_a^3)$, whereas the communication overhead is $O(n_a)$. In short, the running time of the framework can be traded off against the communication overhead for each agent per iteration.

3.4.2 Suboptimality

This section investigates the *suboptimality lower bound* (or can be called *approximation ratio*) of the proposed framework in terms of the global utility, i.e., the objective function in Equation (3.1). Given a partition Π, the global utility value can be equivalently rewritten as

$$J = \sum_{\forall a_i \in \mathcal{A}} u_i(t_{\Pi(i)}, |\mathcal{S}_{\Pi(i)}|). \tag{3.6}$$

Note that we can simply derive $\{x_{ij}\}$ for Equation (3.1) from Π for Equation (3.6), and vice versa. Let J_{GRAPE} and J_{OPT} represent the global utility of a Nash stable partition obtained by the proposed framework and the optimal value, respectively. This chapter refers to the fraction of J_{GRAPE} with respect to J_{OPT} as the *suboptimality* of GRAPE, denoted by α, i.e.,

$$\alpha := J_{GRAPE}/J_{OPT}. \tag{3.7}$$

The lower bound of the suboptimality can be determined by the following theorem.

Theorem 3 (*Suboptimality lower bound: general case*)**.** *Given a Nash stable partition* Π *obtained by the proposed framework, its suboptimality in terms of the global utility is lower bounded as follows:*

$$\alpha \geq J_{GRAPE}/(J_{GRAPE} + \lambda), \tag{3.8}$$

where

$$\lambda \equiv \sum_{\forall \mathcal{S}_j \in \Pi} \max_{a_i \in \mathcal{A}, p \in \{1,\dots,n_a\}} \left\{ p \cdot \left[u_i(t_j, p) - u_i(t_j, |\mathcal{S}_j \cup \{a_i\}|) \right] \right\} \tag{3.9}$$

Proof. Let Π^* denote the optimal partition for the objective function in Equation (3.6). Given a Nash stable partition Π, from Definition 3, it holds that $\forall a_i \in A$

$$u_i(t_{\Pi(i)}, |\mathcal{S}_{\Pi(i)}|) \geq u_i(t^*_{j \leftarrow i}, |\mathcal{S}_j \cup \{a_i\}|), \tag{3.10}$$

where $t^*_{j \leftarrow i}$ indicates task $t_j \in \mathcal{T}$ to which agent a_i should have joined according to the optimal partition Π^*; and $\mathcal{S}_j \in \Pi$ is the coalition for task t_j whose participants follow the Nash stable partition Π.

The right-hand side of the inequality in Equation (3.10) can be rewritten as

$$u_i(t^*_{j \leftarrow i}, |\mathcal{S}_j \cup \{a_i\}|) = u_i(t^*_{j \leftarrow i}, |S^*_j|) - \left\{ u_i(t^*_{j \leftarrow i}, |S^*_j|) - u_i(t^*_{j \leftarrow i}, |\mathcal{S}_j \cup \{a_i\}|) \right\}, \tag{3.11}$$

where $S^*_j \in \Pi^*$ is the ideal coalition of task $t^*_{j \leftarrow i}$ that maximises the objective function.

By summing over all the agents, the inequality in Equation (3.10) can be said that

$$\begin{aligned} &\sum_{\forall a_i \in \mathcal{A}} u_i(t_{\Pi(i)}, |\mathcal{S}_{\Pi(i)}|) \geq \\ &\qquad \sum_{\forall a_i \in \mathcal{A}} u_i(t^*_{j \leftarrow i}, |\mathcal{S}^*_j|) - \sum_{\forall a_i \in \mathcal{A}} \left\{ u_i(t^*_{j \leftarrow i}, |\mathcal{S}^*_j|) - u_i(t^*_{j \leftarrow i}, |\mathcal{S}_j \cup \{a_i\}|) \right\} \end{aligned} \tag{3.12}$$

The left-hand side of the inequality in Equation (3.12) represents the objective function value of the Nash stable partition Π, i.e., J_{GRAPE}, and the first term of the right-hand side is the optimal objective function value, i.e., J_{OPT}. The second term in the right-hand side can be interpreted as the summation of the utility lost of each agent caused by the belated decision to its optimal task, provided that the other agents still follow the Nash stable partition.

The upper bound of the second term is given by

$$\sum_{j=1}^{n_t} |\mathcal{S}_j^*| \cdot \max_{a_i \in \mathcal{S}_j^*} \big\{ u_i(t^*_{j \leftarrow i}, |\mathcal{S}_j^*|) - u_i(t^*_{j \leftarrow i}, |\mathcal{S}_j \cup \{a_i\}|) \big\}. \tag{3.13}$$

This is at most

$$\sum_{\forall \mathcal{S}_j \in \Pi} \max_{a_i \in \mathcal{A}, p \in \{1,\ldots,n_a\}} L_{ij}[p] \equiv \lambda, \tag{3.14}$$

where $L_{ij}[p] = p \cdot \big[u_i(t_j, p) - u_i(t_j, |\mathcal{S}_j \cup \{a_i\}|) \big]$. Note that $u_i(t_0, p)$ was defined to be zero regardless of p.

Hence, the inequality in Equation (3.12) can be rewritten as

$$J_{GRAPE} \geq J_{OPT} - \lambda.$$

Dividing both sides by J_{GRAPE} and rearranging them yield the suboptimality lower bound of the Nash stable partition, as given by Equation (3.8). □

Although Theorem 3 does not provide a fixed-value lower bound, it can be determined as long as a Nash stable partition and agents' individual utility functions are given. Nevertheless, as a special case, if the social utility for any coalition is non-decreasing (or monotonically increasing) in terms of the number of co-working agents, then we can obtain a fixed-value lower bound for the suboptimality of a Nash stable partition.

Theorem 4 (*Suboptimality lower bound: a special case*)**.** *Given an instance* $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ *of GRAPE, if (i) the social utility for any coalition is non-decreasing with regard to the number of participants, i.e., for any* $\mathcal{S}_j \subseteq \mathcal{A}$ *and* $a_l \in \mathcal{A} \setminus \mathcal{S}_j$*, it holds that*

$$\sum_{\forall a_i \in \mathcal{S}_j} u_i(t_j, |\mathcal{S}_j|) \leq \sum_{\forall a_i \in \mathcal{S}_j \cup \{a_l\}} u_i(t_j, |\mathcal{S}_j \cup \{a_l\}|),$$

and (ii) all the individual utilities hold SPAO preference relations, then a Nash stable partition Π *obtained by the proposed framework provides at least* 50% *of suboptimality in terms of the global utility.*

Proof. Firstly, we introduce some definitions and notations that facilitate to describe this proof. Given a partition Π of an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$, the global utility is denoted

by

$$V(\Pi) := \sum_{\forall a_i \in \mathcal{A}} u_i(t_{\Pi(i)}, |\mathcal{S}_{\Pi(i)}|). \tag{3.15}$$

We use the operator $\oplus$ as follows. Given any two partitions $\Pi^A = \{\mathcal{S}_0^A, ..., \mathcal{S}_{n_t}^A\}$ and $\Pi^B = \{\mathcal{S}_0^B, ..., \mathcal{S}_{n_t}^B\}$,

$$\Pi^A \oplus \Pi^B := \{\mathcal{S}_0^A \cup \mathcal{S}_0^B,\ \mathcal{S}_1^A \cup \mathcal{S}_1^B, ...,\ \mathcal{S}_{n_t}^A \cup \mathcal{S}_{n_t}^B\}.$$

Since $\cup_{j=0}^{n_t} \mathcal{S}_j^A = \cup_{j=0}^{n_t} \mathcal{S}_j^B = \mathcal{A}$, there may exist the same agent a_i even in two different coalitions in $\Pi^A \oplus \Pi^B$. For instance, suppose that $\Pi^A = \{\{a_1\}, \{a_2\}, \{a_3\}\}$ and $\Pi^B = \{\emptyset, \{a_1, a_3\}, \{a_2\}\}$. Then, $\Pi^A \oplus \Pi^B = \{\{a_1\}, \{a_1, a_2, a_3\}, \{a_2, a_3\}\}$. We regard such an agent as two different agents in $\Pi^A \oplus \Pi^B$. Accordingly, the operation $\oplus$ may increase the number of total agents in the resultant partition.

Using the definitions described above, the condition (i) implies that

$$V(\Pi^A) \leq V(\Pi^A \oplus \Pi^B). \tag{3.16}$$

From now on, we will show that $\frac{1}{2}V(\Pi^*) \leq V(\hat{\Pi})$, where $\Pi^* = \{\mathcal{S}_0^*, \mathcal{S}_1^*, ..., \mathcal{S}_{n_t}^*\}$ is an optimal partition and $\hat{\Pi} = \{\hat{\mathcal{S}}_0, \hat{\mathcal{S}}_1, ..., \hat{\mathcal{S}}_{n_t}\}$ is a Nash stable partition. By doing so, this theorem can be proved. From the definition in Equation (3.15), it can be said that

$$V(\hat{\Pi} \oplus \Pi^*) = \sum_{\forall a_i \in \mathcal{A}} u_i(t_{\hat{\Pi}(i)}, |\hat{\mathcal{S}}_{\hat{\Pi}(i)} \cup \mathcal{S}^*_{\hat{\Pi}(i)}|) + \sum_{\forall a_i \in \mathcal{A}^-} u_i(t_{\Pi^*(i)}, |\hat{\mathcal{S}}_{\Pi^*(i)} \cup \mathcal{S}^*_{\Pi^*(i)}|), \tag{3.17}$$

where $\mathcal{A}^-$ is the set of agents whose decisions follow not the Nash stable partition $\hat{\Pi}$ but only the optimal partition Π^*. Due to the condition (ii), the first term of the right-hand side in Equation (3.17) is no more than

$$\sum_{\forall a_i \in \mathcal{A}} u_i(t_{\hat{\Pi}(i)}, |\hat{\mathcal{S}}_{\hat{\Pi}(i)}|) \equiv V(\hat{\Pi}). \tag{3.18}$$

Likewise, the second term is also at most

$$\sum_{\forall a_i \in \mathcal{A}^-} u_i(t_{\Pi^*(i)}, |\hat{\mathcal{S}}_{\Pi^*(i)} \cup \{a_i\}|). \tag{3.19}$$

By the definition of Nash stability (i.e., for every agent $a_i \in \mathcal{A}$, $u_i(t_{\hat{\Pi}(i)}, |\hat{\mathcal{S}}_{\hat{\Pi}(i)}|) \geq u_i(t_j, |\hat{\mathcal{S}}_j \cup \{a_i\}|)$, $\forall \hat{\mathcal{S}}_j \in \hat{\Pi}$), the above equation is at most

$$\sum_{\forall a_i \in \mathcal{A}^-} u_i(t_{\hat{\Pi}(i)}, |\hat{\mathcal{S}}_{\hat{\Pi}(i)}|), \tag{3.20}$$

which is also no more than, because of $\mathcal{A}^- \subseteq \mathcal{A}$,

$$\sum_{\forall a_i \in \mathcal{A}} u_i(t_{\hat{\Pi}(i)}, |\hat{\mathcal{S}}_{\hat{\Pi}(i)}|) \equiv V(\hat{\Pi}). \tag{3.21}$$

Accordingly, the left-hand side of Equation (3.17) holds the following inequality:

$$V(\hat{\Pi} \oplus \Pi^*) \leq 2V(\hat{\Pi}). \tag{3.22}$$

Thanks to Equation (3.16), it follows that

$$V(\Pi^*) \leq V(\hat{\Pi} \oplus \Pi^*).$$

Therefore, $V(\Pi^*) \leq 2V(\hat{\Pi})$, which completes this proof. □

3.4.3 Adaptability

Our proposed framework is also expected to be adaptable to dynamic environments such as unexpected addition or loss of agents or tasks, owing to its fast convergence to a Nash stable partition. Thanks to Lemma 1, if a new agent additionally joins an ongoing mission in which an agreed assignment was already determined, the number of iterations required for converging to a new Nash stable partition is at most the number of the total agents. Responding to any environmental change, the framework is able to establish a new agreed task assignment within polynomial time.

3.4.4 Robustness in Asynchronous Environments

In the proposed framework, for every iteration, each agent does not need to wait until nor ensure that its locally-known information has been propagated to a certain neighbor

group. Instead, as described in Remark 5, it is enough for the agent to receive the local information from one of its neighbors, to make a decision, and to send the updated partition back to some of its neighbors. Temporary disconnection or non-operation of some agents may cause dummy iterations additionally. However, it does not affect the existence of, the convergence toward, and the suboptimality of a Nash stable partition under the proposed framework, which is also supported by Section 3.6.5.

The information-sharing requirement for GRAPE is weaker than *Global Information Consistency Assumption (GICA)* (i.e., global information needs to be consistently known by the entire agents at each iteration) or *Local Information Consistency Assumption (LICA)* (i.e., local information needs to be consistently known by a local agent group at each iteration) [42]. In this thesis, we refer to this requirement as *Neighbour Information Consistency Assumption (NICA)* (i.e., an agent's local information only needs to be known by one of its neighbour agents at each iteration).

3.5 GRAPE with Minimum Requirements

This section addresses another task allocation problem where each task may require at least a certain number of agents for its completion. This problem can be defined as follows.

Problem 2. Given a set of agents $\mathcal{A}$ and a set of tasks $\mathcal{T}$, the objective is to find an assignment $\{x_{ij}\}$ such that

$$\max_{\{x_{ij}\}} \sum_{\forall a_i \in \mathcal{A}} \sum_{\forall t_j \in \mathcal{T}} u_i(t_j, p_j) x_{ij}, \tag{3.23}$$

where for every t_j

$$p_j = \sum_{\forall a_i \in \mathcal{A}} x_{ij},$$

subject to

$$\sum_{\forall a_i \in \mathcal{A}} x_{ij} \geq R_j \quad \forall t_j \in \mathcal{T}, \tag{3.24}$$

$$\sum_{\forall t_j \in \mathcal{T}} x_{ij} \leq 1 \quad \forall a_i \in \mathcal{A}, \tag{3.25}$$

$$x_{ij} \in \{0,1\} \quad \forall a_i \in \mathcal{A}, \forall t_j \in \mathcal{T}. \tag{3.26}$$

Here, $R_j \in \mathbb{N} \cup \{0\}$ is the number of minimum required agents for task t_j, and all the other variables are identically defined as those in Problem 1. It is considered that, for $\forall a_i \in \mathcal{A}$ and $\forall t_j \in \mathcal{T}$,

$$u_i(t_j, p_j) = 0 \quad \text{if } p_j < R_j, \tag{3.27}$$

because task t_j cannot be completed in this case. Note that any task t_j without such a requirement is regarded to have $R_j = 0$.

For each task t_j having $R_j > 0$, even if $u_i(t_j, p_j)$ is monotonically decreasing when $p_j \geq R_j$, the individual utility can not be simply transformed to a preference relation holding SPAO because of Equation (3.27). Thus, we need to modify the utility function to yield alternative values for the case when $p_j < R_j$. We refer to the modified utility as *auxiliary individual utility* $\tilde{u}_i$, which is defined as

$$\tilde{u}_i(t_j, p_j) := \begin{cases} u_i^0(t_j, p_j) & \text{if } p_j \leq R_j, \\ u_i(t_j, p_j) & \text{otherwise,} \end{cases} \tag{3.28}$$

where $u_i^0(t_j, p_j)$ is the *dummy utility* of agent a_i with regard to task t_j when $p_j \leq R_j$. The dummy utility is intentionally used also for the case when $p_j = R_j$. The value of $\tilde{u}_i(t_j, R_j)$ should satisfy the following condition.

Condition 1. For every agent $a_i \in \mathcal{A}$, its preference relation $\mathcal{P}_i$ holds that

$$(t_j, R_j) \succ_i (t_k, R_k + 1) \qquad \forall t_j, t_k \in \mathcal{T}.$$

This condition enables every agent to prefer a task for which the number of co-working agents is less than its minimum requirement, over any other tasks whose requirements are already fulfilled. Under this condition, if the agent set $\mathcal{A}$ is such that $|\mathcal{A}| \geq \sum_{\forall t_j \in \mathcal{T}} R_j$ and a Nash stable partition is found, then the resultant assignment satisfies Equation (3.24).

Proposition 1. *Given an instance of Problem 2 where $u_i(t_j, p_j)$ $\forall i$ $\forall j$ is a monotonically decreasing function with regard to $\forall p_j \geq R_j$, if the dummy utilities $u_i^0(t_j, p_j)$ $\forall i$ $\forall j$*

in Equation (3.28) are set to satisfy Condition 1 and SPAO for $\forall p_j \leq R_j$*, then all the resultant auxiliary individual utilities* $\tilde{u}_i(t_j, p_j)$ $\forall i$ $\forall j$ $\forall p_j$ *can be transformed to a* n_a*-tuple of preference relations* $\mathcal{P}$ *that hold Condition 1 as well as SPAO for* $\forall p_j \in \{1, ..., n_a\}$*. In the corresponding instance of GRAPE* $(\mathcal{A}, \mathcal{T}, \mathcal{P})$*, a Nash stable partition can be determined within polynomial times as shown in Theorems 1 and 2 because of SPAO, and the resultant partition can satisfy Equation (3.24) due to Condition 1.*

Let us give an example. Suppose that there exist 100 agents $\mathcal{A}$, and 3 tasks $\mathcal{T} = \{t_1, t_2, t_3\}$ where only t_3 has its minimum requirement $R_3 = 5$; for every agent $a_i \in \mathcal{A}$, individual utilities for t_1 and t_2, i.e., $u_i(t_1, p)$ and $u_i(t_1, p)$, are much higher than that for t_3 in $\forall p \in \{1, ..., 100\}$. We can find a Nash stable partition for this example, as described in Proposition 1, by setting $u_i^0(t_3, p) := \max_{\forall t_j}\{u_i(t_j, R_j + 1)\} + \beta$ for $\forall p \leq R_3, \forall a_i \in \mathcal{A}$, where $\beta > 0$ is an arbitrary positive constant.

After a Nash stable partition is found, in order to compute the objective function value in (3.23), the original individual utility function u_i should be used instead of the auxiliary one $\tilde{u}_i$.

Proposition 2. *Given a Nash stable partition* Π *obtained by implementing Proposition 1, its suboptimality bound* α *is such that*

$$\alpha \geq \frac{J_{GRAPE}}{J_{GRAPE} + \tilde{\lambda}} \cdot \frac{J_{GRAPE}}{J_{GRAPE} + \delta}. \tag{3.29}$$

Here, $\delta := \tilde{J}_{GRAPE} - J_{GRAPE}$*, where* $\tilde{J}_{GRAPE}$ *(or* J_{GRAPE}*) is the objective function value in (3.23) using* $\tilde{u}_i$ *(or using* u_i*) given the Nash stable partition. Likewise,* $\tilde{\lambda}$ *is the value in (3.9) using* $\tilde{u}_i$*. In addition to this, if every* $\tilde{u}_i$ *satisfies the conditions for Theorem 4, then*

$$\alpha \geq \frac{1}{2} \cdot \frac{J_{GRAPE}}{J_{GRAPE} + \delta}. \tag{3.30}$$

Proof. Since the Nash stable partition Π is obtained by using $\tilde{u}_i$, it can be said from Equations (3.7) and (3.8) that

$$\frac{\tilde{J}_{GRAPE}}{\tilde{J}_{OPT}} \geq \frac{\tilde{J}_{GRAPE}}{\tilde{J}_{GRAPE} + \tilde{\lambda}}. \tag{3.31}$$

Due to the fact that $\tilde{u}_i(t_j, p_j) \geq u_i(t_j, p_j)$ for $\forall i, j, p_j$, it is clear that $\tilde{J}_{GRAPE} \geq J_{GRAPE}$ and $\tilde{J}_{OPT} \geq J_{OPT}$. By letting that $\delta := \tilde{J}_{GRAPE} - J_{GRAPE}$, the left term in (3.31) is at

most $(J_{GRAPE}+\delta)/J_{OPT}$. Besides, the right term in (3.31) is a monotonically-increasing function with regard to $\tilde{J}_{GRAPE}$, and thus, it is lower bounded by $\frac{J_{GRAPE}}{(J_{GRAPE}+\tilde{\lambda})}$. From this, Equation (3.31) can be rewritten as Equation (3.29) by multiplying $\frac{J_{GRAPE}}{(J_{GRAPE}+\delta)}$.

Likewise, for the case when every $\tilde{u}_i$ satisfies the conditions for Theorem 4, it can be said that $\tilde{J}_{GRAPE} \geq 1/2 \cdot \tilde{J}_{OPT}$, which can be transformed into Equation (3.30) as shown above. □

Notice that if $\delta = 0$ for the Nash stable partition in Proposition 2, then the suboptimality bounds become equivalent to those in Theorems 3 and 4.

3.6 Simulation and Results

This section validates the performances of the proposed framework with respect to its scalability, suboptimality, adaptability against dynamic environments, and robustness in asynchronous environments.

3.6.1 Mission Scenario and Settings

3.6.1.1 Utility functions

Firstly, we introduce the social and individual utilities used in this numerical experiment. We consider that if multiple robots execute a task together as a coalition, then they are given a certain level of reward for the task. The amount of the reward varies depending on the number of the co-working agents. The reward is shared with the agents, and each agent's individual utility is considered as the shared reward minus the cost required to personally spend on the task (e.g., fuel consumption for movement). In this experiment, *the equal fair allocation rule* [43, 44] is adopted. Under the rule, a task's reward is equally shared amongst the members. Therefore, the individual utility of agent a_i executing task t_j with coalition $\mathcal{S}_j$ is defined as

$$u_i(t_j, |\mathcal{S}_j|) = \mathsf{r}(t_j, |\mathcal{S}_j|)/|\mathcal{S}_j| - \mathsf{c}_i(t_j), \tag{3.32}$$

where $\mathsf{r}(t_j, |\mathcal{S}_j|)$ is the reward from task t_j when it is executed by $\mathcal{S}_j$ together, and $\mathsf{c}_i(t_j)$ is the cost that agent a_i needs to pay for the task. Here, we simply set the cost as

a function of the distance from agent a_i to task t_j. We set that if $u_i(t_j, |\mathcal{S}_j|)$ is not positive, agent a_i prefers to join $\mathcal{S}_0$ over $\mathcal{S}_j$.

This experiment considers two types of tasks. For the first type, a task's reward becomes higher as the number of participants gets close to a specific desired number. We refer to such a task as a *peaked-reward* task, and its reward can be defined as

$$\mathsf{r}(t_j, |\mathcal{S}_j|) = \frac{\mathsf{r}_j^{\max} \cdot |\mathcal{S}_j|}{n_j^d} \cdot e^{-|\mathcal{S}_j|/n_j^d + 1}, \tag{3.33}$$

where n_j^d represents the desired number, and $\mathsf{r}_j^{\max}$ is the peaked reward in case that n_j^d of agents are involved in. Consequently, the individual utility of agent a_i with regard to task t_j becomes the following equation:

$$u_i(t_j, |\mathcal{S}_j|) = \frac{\mathsf{r}_j^{\max}}{n_j^d} \cdot e^{-|\mathcal{S}_j|/n_j^d + 1} - \mathsf{c}_i(t_j). \tag{3.34}$$

For the second type, a task's reward becomes higher as more agents are involved, but the corresponding marginal gain decreases. This type of tasks is said to be *(monotone) submodular-reward*, and the reward can be defined as

$$r(t_j, |\mathcal{S}_j|) = \mathsf{r}_j^{\min} \cdot \log_{\epsilon_j}(|\mathcal{S}_j| + \epsilon_j - 1), \tag{3.35}$$

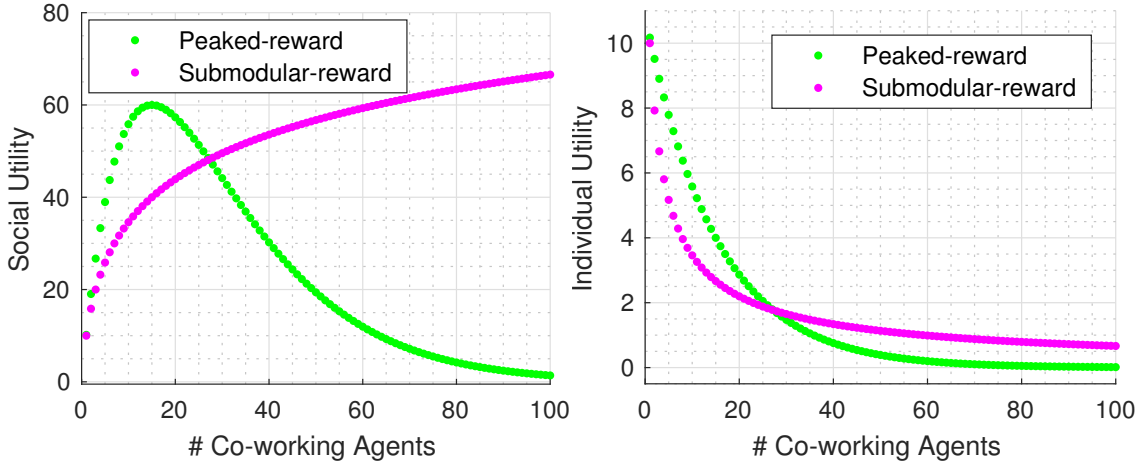
where $\mathsf{r}_j^{\min}$ indicates the reward obtained if there is only one agent involved, and $\epsilon_j > 1$ is the design parameter regarding the diminishing marginal gain. The resultant individual utility becomes as follows:

$$u_i(t_j, |\mathcal{S}_j|) = \mathsf{r}_j^{\min} \cdot \log_{\epsilon_j}(|\mathcal{S}_j| + \epsilon_j - 1)/|\mathcal{S}_j| - \mathsf{c}_i(t_j). \tag{3.36}$$

Figure 3.1 illustrates examples of the social utilities and individual utilities for the task types introduced above. For simplification, agents' costs are ignored in the figure. We set $\mathsf{r}_j^{\max}$, n_j^d, $\mathsf{r}_j^{\min}$ and ϵ_j to be 60, 15, 10, and 2, respectively. Notice that the individual utilities are monotonically decreasing in both cases, as depicted in Figure 3.1(b). Therefore, given a mission that entails these task types, we can generate an instance $(\mathcal{A}, \mathcal{T}, \mathcal{P})$ of GRAPE that holds SPAO.

3.6.1.2 Parameters generation

In the following sections, we will mainly utilise Monte Carlo simulations. At each run, n_t tasks and n_a agents are uniform-randomly located in a 1000 $m \times 1000$ m arena and a



(a) Social utility **(b)** Individual utility

Figure 3.1: Examples of utility functions used in the numerical experiment are shown, depending on the two different task types (i.e., peaked-reward and submodular-reward): **(a)** the social utility of a coalition; **(b)** an agent's individual utility.

250 $m \times 250$ m arena within there, respectively. For a scenario including peaked-reward tasks, $\mathsf{r}_j^{\max}$ is randomly generated from a uniform distribution over $[1000, 2000] \times n_a/n_t$, and n_j^d is set to be the rounded value of $(\mathsf{r}_j^{\max}/\sum_{\forall t_k \in \mathcal{T}^*} \mathsf{r}_k^{\max}) \times n_a$. For a scenario including submodular-reward tasks, ϵ_j is set as 2, and $\mathsf{r}_j^{\min}$ is uniform-randomly generated over $[1000, 2000] \times 1/\log_{\epsilon_j}(n_a/n_t + 1)$.

3.6.1.3 Communication network

Given a set of agents, their communication network is strongly-connected in a way that only contains a bidirectional minimum spanning tree with consideration of the agents' positions. Furthermore, we also consider the fully-connected network in some experiments in order to examine the influence of the network. The communication network is randomly generated at each instance, and is assumed to be sustained during a mission except the robustness test simulations in Section 3.6.5.

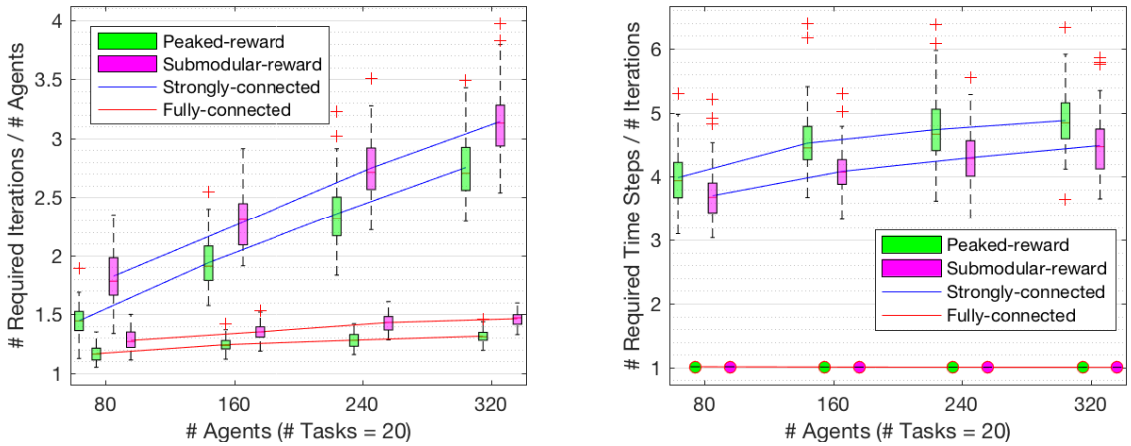
3.6.2 Scalability

To investigate the effectiveness of n_t and n_a upon the scalability of the proposed approach, we conduct a Monte Carlo simulation with 100 runs for the scenarios introduced in Section 3.6.1 with a fixed $n_t = 20$ and various $n_a \in \{80, 160, 240, 320\}$ and for those with $n_a = 160$ and $n_t \in \{5, 10, 15, 20\}$. Figure 3.2 shows the statistical results using box-and-whisker plots, where the green boxes indicate the results from the scenarios with the peaked-reward tasks and the magenta boxes are those with the submodular-reward tasks. The blue and red lines connecting the boxes represent the average value for each test case (n_a, n_t) under a strongly-connected network and the fully-connected network, respectively.

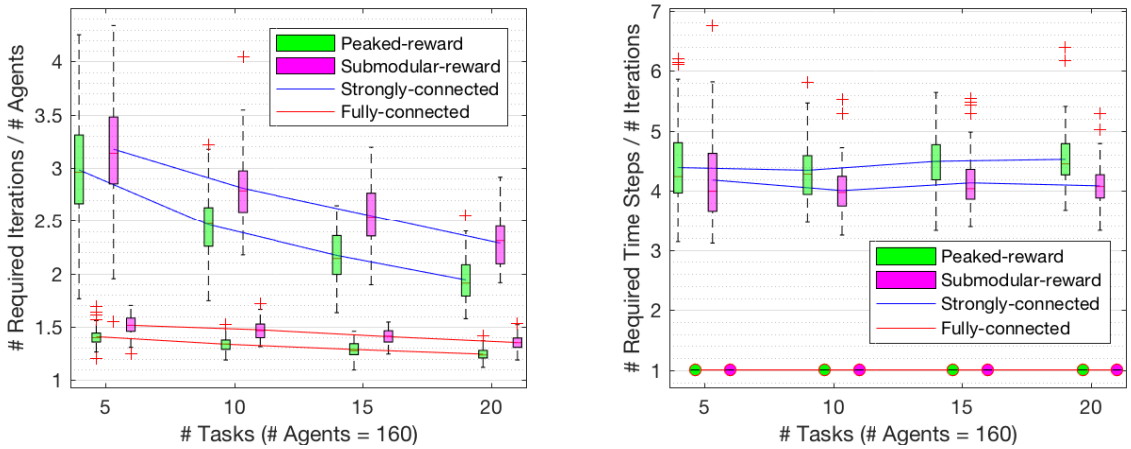
The left subfigure in Figure 3.2(a) shows that the ratio of the number of required (normal) iterations to that of agents linearly increases as more agents are involved. This implies that the proposed framework has quadratic complexity with regard to the number of agents (i.e., $C_1 n_a^2$), as stated in Theorem 2, but with C_1 being much less than $\frac{1}{2}$, which is the value from the theorem. Even C_1 can become even lower (e.g., $C_1 = 5 \times 10^{-4}$ in the experiments) under the fully-connected network. Such a C_1 being smaller than $\frac{1}{2}$ may be explained by Remark 4: the algorithmic efficiency of Algorithm 1 can reduce unnecessary iterations that may be induced in the procedure of the proof for Theorem 2.

On the other hand, the left subfigure in Figure 3.2(b) shows that the number of required iterations decreases with regard to the number of tasks. This trend may be caused by the fact that more selectable options provided to the fixed number of agents can reduce possible conflicts between the agents.

Furthermore, in the two results, the trends regarding either n_a or n_t have higher slopes under a strongly-connected network than those under the fully-connected network. This is because the former condition is more sensitive to conflicts between agents, and thus causes additional iterations. For example, agents at the middle nodes of the network may change their decisions (and thus increase the number of iterations) while the local partition information of the agent at one end node is being propagated to agents at the other end nodes. Such unnecessary iterations in the middle might not have occurred if the agents at all the end nodes were directly connected to each other.



(a) $n_a \in \{80, 160, 240, 320\}$ with fixed $n_t = 20$



(b) $n_t \in \{5, 10, 15, 20\}$ with fixed $n_a = 160$

Figure 3.2: Convergence performance of the proposed framework is shown, depending on communication networks (i.e., Strongly-connected vs. Fully-connected) and utility function types (i.e., Peaked-reward vs. Submodular-reward) with different number of agents and tasks: **(Left)** the number of (normal) iterations happened relative to that of agents; **(Right)** the number of time steps happened (i.e., normal and dummy iterations) relative to that of iterations.

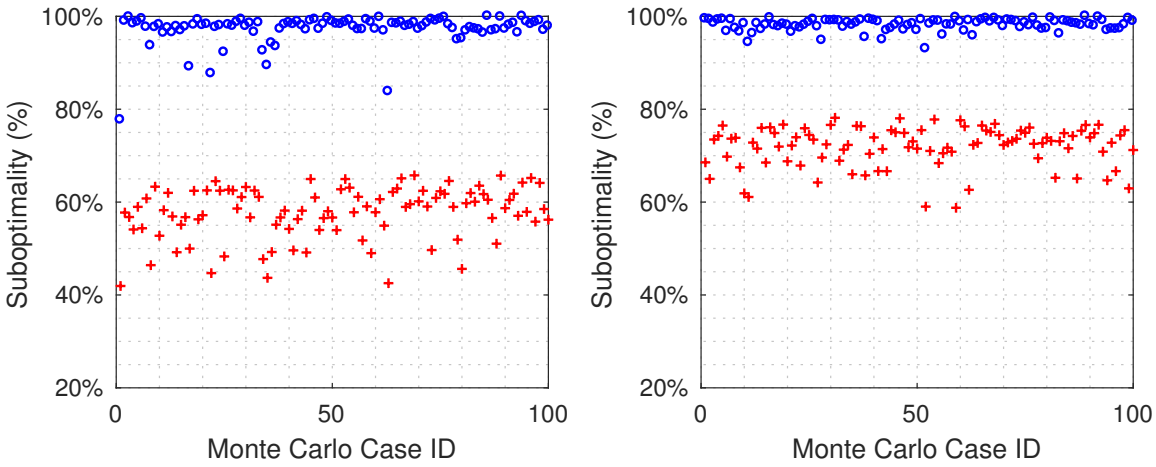
The right subfigures in Figure 3.2(a) and (b) indicate that approximately 3–4 times of dummy iterations, compared with the required number of normal iterations, are additionally needed under a strongly-connected network. Noting that the mean values of the graph diameter d_G for the instances with $n_a \in \{80, 160, 240, 320\}$ are $36, 58, 75$ and 92, respectively, the results show that the amount of dummy iterations happened is averagely much less than the bound value, which is d_G as pointed out in Section 3.4.1. On the contrary, under the fully-connected network there is no need of such a dummy iteration, and thus the required number of iterations and that of time steps are the same.

3.6.3 Suboptimality

This section examines the suboptimality of the proposed framework by using Monte Carlo simulations with 100 instances. In each instance, there are $n_t = 3$ of tasks and $n_a = 12$ of agents who are strongly-connected. Figure 3.3 presents the true suboptimality of each instance, which is the ratio of the global utility obtained by the proposed framework to that by a brute-force search, i.e., J_{GRAPE}/J_{OPT}, and the lower bound given by Theorem 3. A blue circle and a red cross in the figure indicate the true suboptimality and the lower bound, respectively. The results show that the framework provides near-optimal solutions in almost all cases and the suboptimality of each Nash stable partition is enclosed by the corresponding lower bound.

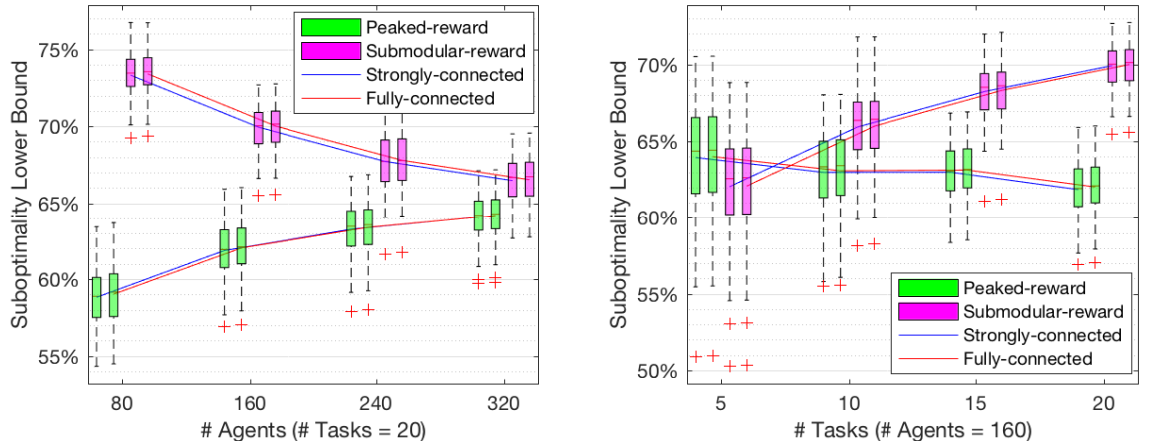
The suboptimality may be improved if the agents are let to investigate a larger search space, for example, possible coalitions caused by co-deviation of multiple agents. However, this strategy in return may increase communication transactions between the agents because they have to notice each other's willingness unless their individual utility functions are known to each other, which is in contradiction to Assumption 4. Besides, the computational overhead for each agent per iteration also becomes more expensive than $O(n_t)$, which is the complexity bound for unilateral searching, as shown in Section 3.4.1. Hence, the resultant algorithm's complexity may hinder its practical applicability to a large-scale multiple agent system, which is believed to be more critical rather than global optimality in the domain [45–47].

Figure 3.4 depicts the suboptimality lower bounds for the large-size problems that



(a) Peaked-reward task **(b)** Submodular-reward task

Figure 3.3: True suboptimality of a Nash stable partition obtained by GRAPE for each run of the Monte Carlo simulation (denoted by a blue circle) and its lower bound provided by Theorem 3 (denoted by a red cross) under a strongly-connected communication network: **(a)** the scenarios with peaked-reward tasks; **(b)** the scenarios with submodular-reward tasks



(a) **(b)**

Figure 3.4: The suboptimality lower bound, given by Theorem 3, of a Nash stable partition obtained by GRAPE, depending on communication networks (i.e., Strongly-connected vs. Fully-connected) and utility function types (i.e., Peaked-reward vs. Submodular-reward): **(a)** fixed $n_t = 20$ with varying $n_a \in \{80, 160, 240, 320\}$; **(b)** fixed $n_a = 160$ with varying $n_t \in \{5, 10, 15, 20\}$

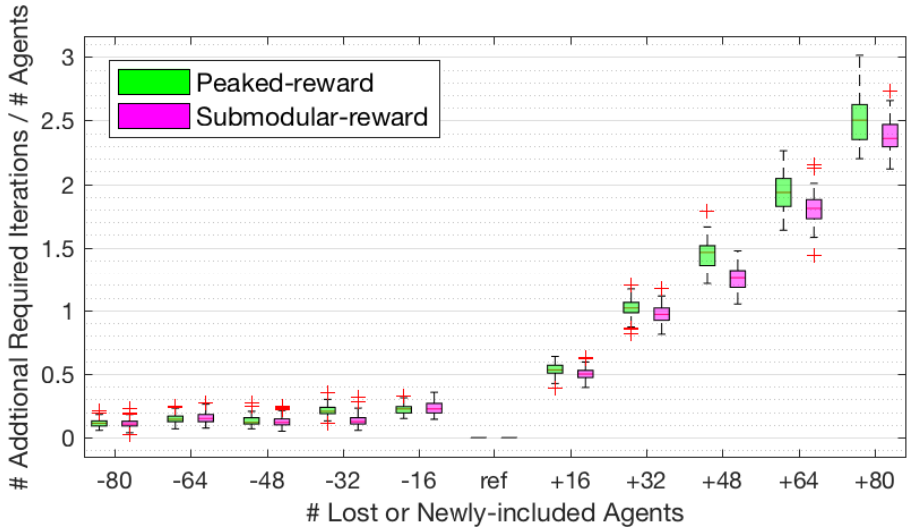
were previously addressed in Section 3.6.2. It is clearly shown that the agent communication network does not make any effect on the suboptimality lower bound of a Nash stable partition. Although there is no universal trend of the suboptimality with regard to n_a and n_t in both utility types, it is suggested that the features of the lower bound given by Theorem 3 can be influenced by the utility functions considered. In the experiments, the suboptimality bound averagely remain above than 60–70 %.

3.6.4 Adaptability

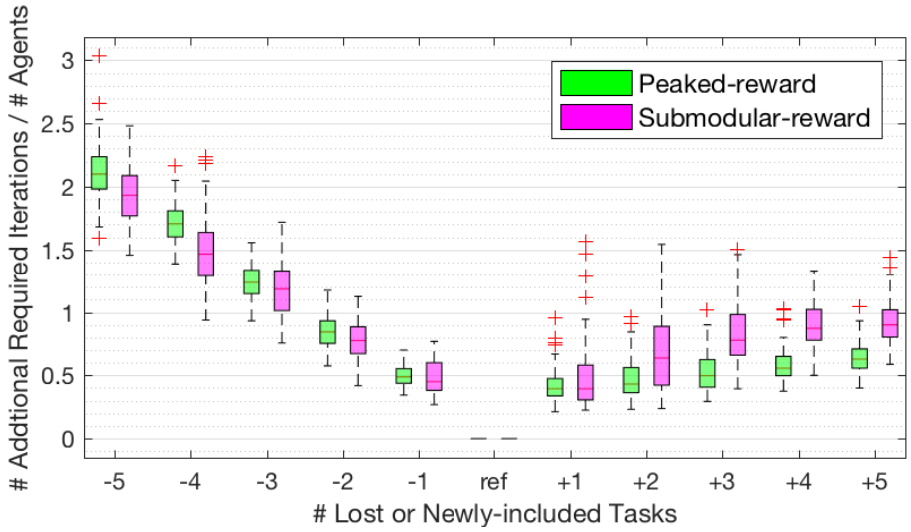
This section discusses the adaptability of our proposed framework in response to dynamic environments such as unexpected inclusion or loss of agents and tasks. Suppose that there are 10 tasks and 160 agents in a mission, and a Nash stable partition was already found as a baseline. During the mission, the number of agents (or tasks) changes; the range of the change is from losing 50% of the existing agents (or tasks) to additionally including new ones as much as 50% of them. For each dynamical environment, a Monte Carlo simulation with 100 instances is performed by randomly including or excluding a subset of the corresponding number of agents or tasks. Here, we consider a strongly-connected communication network.

Figure 3.5(a) illustrates that the more agents are involved additionally, the more iterations are required for converging to a new Nash stable partition. This is because the inclusion of a new agent may lead to additional iterations at most as much as the number of the total agents including the new agent (as shown in Lemma 1). On the contrary, the loss of existing agents does not seem to have any apparent relation with the number of iterations. A possible explanation is that the exclusion of an existing agent is favourable to the other agents due to SPAO preferences. This stimulates only a limited number of agents who are preferred to move to the task-specific coalition where the excluded agent was. This feature induces fewer additional iterations to reach a new Nash stable partition, compared with the case of adding a new agent.

Figure 3.5(b) shows that to eliminate existing tasks causes more iterations than including new tasks. This can be explained by the fact that removing any task releases the agents performing the task free and it results in extra iterations at least the number of the freed agents. On the other hand, adding new tasks induces relatively fewer



(a) Dynamic Agents



(b) Dynamic Tasks

Figure 3.5: The number of additional iterations required for re-converging a Nash stable partition relative to the number of agents in the case when some agents or tasks are partially lost of newly involved (Baseline: $n_t = 10$, $n_a = 160$, and a Nash stable partition was already found). Negative values in the x-axis indicate that the corresponding number of existing agents or tasks are lost. Positive values indicate that the corresponding number of new agents or tasks are included in an ongoing mission. A strongly-connected communication network is used.

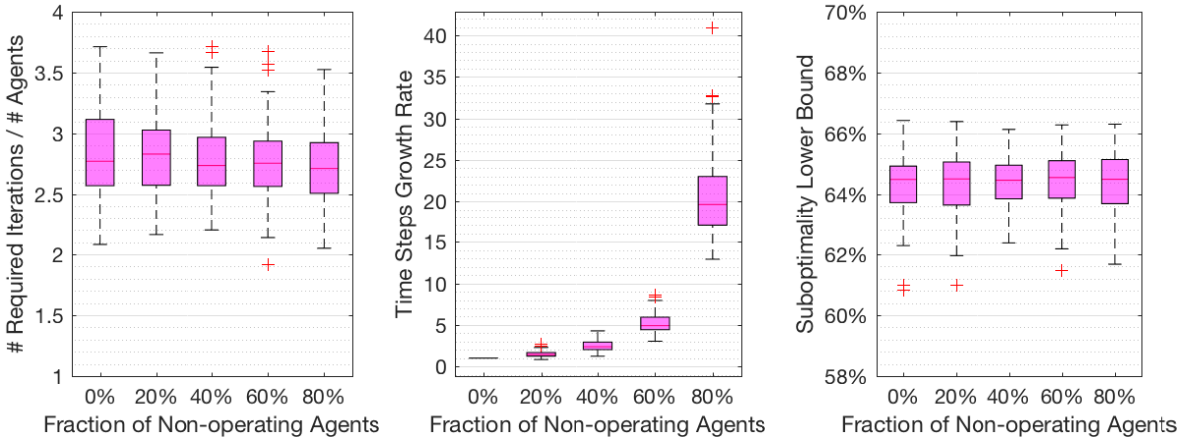
additional iterations, because only some of the existing agents are attracted to these tasks.

In summary, as the ratio of the number of agents to that of tasks increases, the number of additional iterations for convergence towards a new Nash stable partition also increases. This actually corresponds to the trend described in Section 3.6.2, i.e., the left subfigures in Figure 3.2(a) and (b). In all the cases of this experiment, the number of additionally induced iterations still remains at the same order of the number of the given agents, which implies that the proposed framework provides excellent adaptability.

3.6.5 Robustness in Asynchronous Environments

This section investigates the robustness of the proposed framework in asynchronous environments. This scenario assumes that a certain fraction of the given agents, which are randomly chosen at each time step, somehow can not execute Algorithm 1 and even can not communicate with other normally-working neighbour agents. We refer to such agents as *non-operating* agents. Given that $n_t = 5$ and $n_a = 40$, the fractions of the non-operating agents are set as $\{0, 0.2, 0.4, 0.6, 0.8\}$. In each case, we conduct 100 instances of Monte Carlo experiments for which the submodular-reward utilities are used.

Figure 3.6(a) presents that the number of (normal) iterations required for converging towards a Nash stable partition remains the same level regardless of the fraction of the non-operating agents. Despite that, the required time steps increase as more agents become non-operating, as shown in Figure 3.6(b). Note that *time steps growth rate* means the ratio of the total required time steps to those for the case when all the agents operate normally. These findings indicate that, due to communicational discontinuity caused by the non-operating agents, the framework may take more time to wait for these agents to operate again and then to disseminate locally-known partition information over the entire agents. Accordingly, dummy iterations may increase in asynchronous environments, but the proposed framework is still able to find a Nash stable partition. Furthermore, the resultant Nash stable partition's suboptimality lower bound obtained by Theorem 3 is not affected, as presented in Figure 3.6(c).



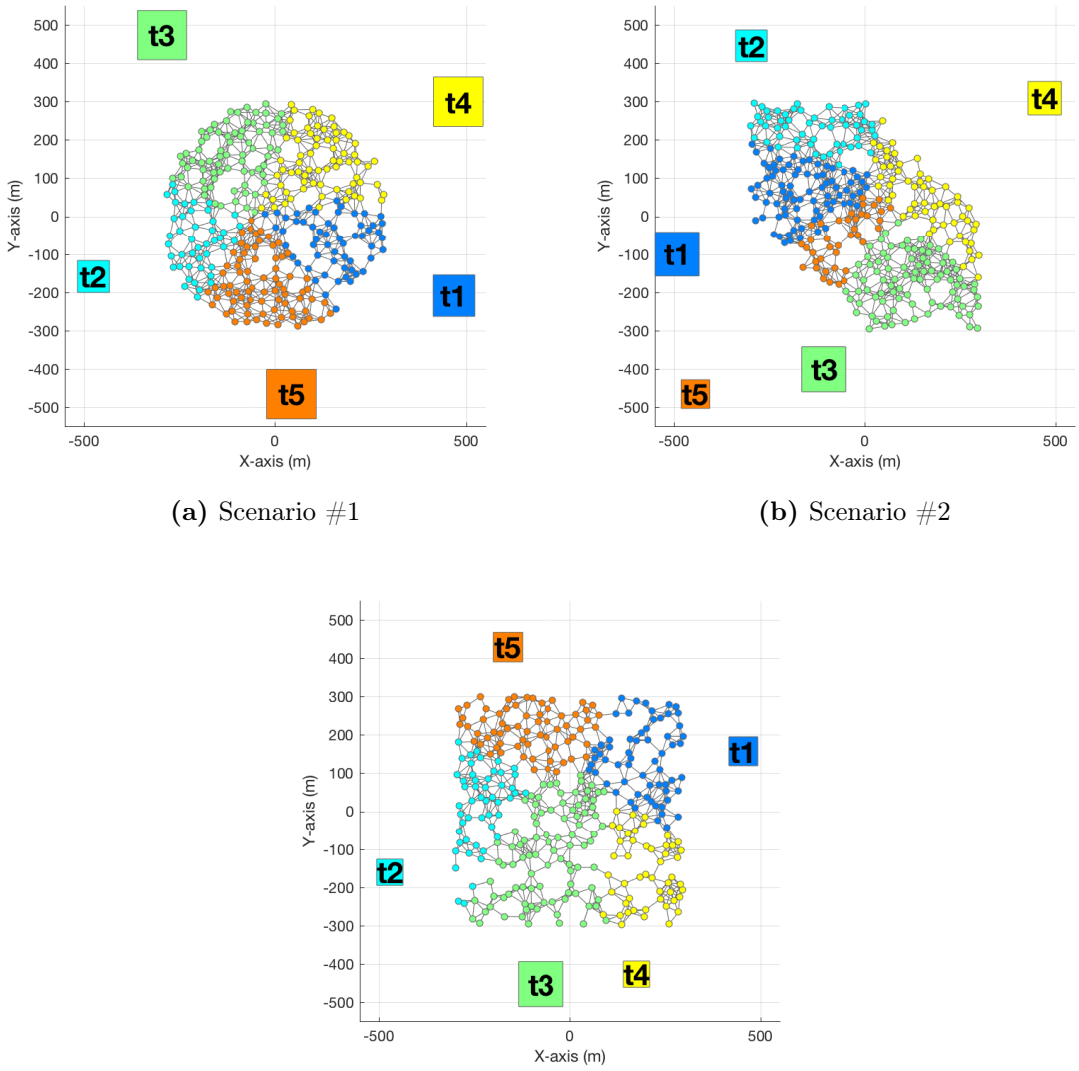
(a) **(b)** **(c)**

Figure 3.6: Robustness test in asynchronous environments at scenarios with $n_t = 10, n_a = 160$, and the submodular-reward tasks: the plot shows the effectiveness of the fraction of non-operating agents with regard to: **(a)** the number of iterations happened until convergence relative to that of agents; **(b)** the ratio of the time steps happened to those for the normal case; **(c)** the suboptimality lower bound by Theorem 3.

3.6.6 Visualisation

We have $n_a = 320$ agents and $n_t = 5$ tasks. The initial locations of the given agents are randomly generated, and the overall formation shape is different in each test scenario such as being circle, skewed circle, and square (denoted by Scenario #1, #2, and #3, respectively). The tasks are also randomly located away from the agents. In this simulation, each agent is able to communicate with its nearby agents within a radius of 50 m. Here, the submodular-reward tasks are used.

Figure 3.7 shows the visualised task allocation results, where the circles and the squares indicate the positions of the agents and the tasks, respectively. The lines between the circles represent the communication networks of the agents. The coloured agents are assigned to the same coloured task, for example, yellow agents belong to the team for executing the yellow task. The size of a square indicates the reward of the corresponding task. The cost for an agent with regard to a task is considered as a function of the distance from the agent to the task. The allocation results seem to



(a) Scenario #1

(b) Scenario #2

(c) Scenario #3

Figure 3.7: Visualised task allocation results with different geographic scenarios ($n_t = 5$, $n_a = 320$). Each square and its size represent each task's position and its reward (or demand), respectively. The circles and the lines between them indicate the positions of agents and their communication network, respectively. The colour of each circle implies that the corresponding agent is assigned to the same coloured task.

be reasonable with consideration of the task rewards and the costs. The number of iterations required to find a Nash stable partition is 1355, 1380, and 1295 for Scenario #1, #2, and #3, respectively. Since the communication networks are more connected than a strongly-connected one, the number of dummy iterations happened is just 20–30% of that of the iterations. This value is much lower than the results shown in Figure 3.2 because the networks considered here are more connected than those in Section 3.6.2.

3.7 Conclusion

This chapter proposed a novel game-theoretical framework that addresses a task allocation problem for a robotic swarm consisting of self-interested agents. We showed that selfish agents whose individual interests are transformable to SPAO preferences can converge to a Nash stable partition by using the proposed simple decentralised algorithm, which is executable even in asynchronous environments and under a strongly-connected communication network. We analytically and experimentally presented that the proposed framework provides scalability, a certain level of guaranteed suboptimality, adaptability, robustness, and the potential to accommodate different interests of agents.

As this framework can be considered as a new sub-branch of self-organised approaches, one of our ongoing works is to compare it with one of the existing methods. Defining a fair scenario for both methods is non-trivial and requires careful consideration; otherwise, a resultant unsuitable scenario may provide biased results. Secondly, another natural progression of this study is to relax anonymity of agents and thus to consider a combination of the agents' identities. Experimentally, we have often observed that heterogeneous agents with social inhibition also can converge to a Nash stable partition. More research would be needed to analyse the quality of a Nash stable partition obtained by the proposed framework in terms of min max optimisation because our various experiments showed that the outcome provides individual utilities to agents in a balanced manner.

References

[1] J. H. Drèze and J. Greenberg, "Hedonic Coalitions: Optimality and Stability," *Econometrica*, vol. 48, no. 4, pp. 987–1003, 1980.

[2] S. Banerjee, H. Konishi, and T. Sönmez, "Core in a Simple Coalition Formation Game," *Social Choice and Welfare*, vol. 18, no. 1, pp. 135–153, 2001.

[3] A. Bogomolnaia and M. O. Jackson, "The Stability of Hedonic Coalition Structures," *Games and Economic Behavior*, vol. 38, no. 2, pp. 201–230, 2002.

[4] B. P. Gerkey and M. J. Matarić, "A Formal Analysis and Taxonomy of Task Allocation in Multi-robot Systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[5] G. A. Korsah, A. Stentz, and M. B. Dias, "A Comprehensive Taxonomy for Multi-robot Task Allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[6] A. Brutschy, G. Pini, C. Pinciroli, M. Birattari, and M. Dorigo, "Self-organized Task Allocation to Sequentially Interdependent Tasks in Swarm Robotics," *Autonomous Agents and Multi-Agent Systems*, vol. 28, no. 1, pp. 101–125, 2014.

[7] N. Kalra and A. Martinoli, "A Comparative Study of Market-Based and Threshold-Based Task Allocation," in *Distributed Autonomous Robotic Systems 7*, Tokyo, Ed. Springer Japan, 2006, pp. 91–101.

[8] Y. Zhang and L. E. Parker, "Considering Inter-task Resource Constraints in Task Allocation," *Autonomous Agents and Multi-Agent Systems*, vol. 26, no. 3, pp. 389–419, 2013.

[9] H. L. Choi, L. Brunet, and J. P. How, "Consensus-based Decentralized Auctions for Robust Task Allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[10] P. Segui-Gasco, H.-S. Shin, A. Tsourdos, and V. J. Segui, "Decentralised Submodular Multi-Robot Task Allocation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hamburg, Germany, 2015, pp. 2829–2834.

[11] B. Acikmese and D. S. Bayard, "Markov Chain Approach to Probabilistic Guidance for Swarms of Autonomous Agents," *Asian Journal of Control*, vol. 17, no. 4, pp. 1105–1124, 2015.

[12] I. Chattopadhyay and A. Ray, "Supervised Self-Organization of Homogeneous Swarms Using Ergodic Projections of Markov Chains," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1505–1515, 2009.

[13] N. Demir and B. Acikmese, "Probabilistic Density Control for Swarm of Decentralized ON-OFF Agents with Safety Constraints," in *American Control Conference*, Chicago, IL, USA, 2015, pp. 5238–5244.

[14] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1103–1123, 2017.

[15] I. Jang, H.-S. Shin, and A. Tsourdos, "Bio-Inspired Local Information-Based Control for Probabilistic Swarm Distribution Guidance," *arXiv:1711.06869 [cs.MA]*, 2017.

[16] S. Berman, A. Halasz, M. A. Hsieh, and V. Kumar, "Optimized Stochastic Policies for Task Allocation in Swarms of Robots," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 927–937, 2009.

[17] A. Halasz, M. A. Hsieh, S. Berman, and V. Kumar, "Dynamic Redistribution of a Swarm of Robots among Multiple Sites," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, 2007, pp. 2320–2325.

[18] M. A. Hsieh, A. Halasz, S. Berman, and V. Kumar, "Biologically Inspired Redistribution of a Swarm of Robots among Multiple Sites," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 121–141, 2008.

[19] T. W. Mather and M. A. Hsieh, "Macroscopic Modeling of Stochastic Deployment Policies with Time Delays for Robot Ensembles," *The International Journal of Robotics Research*, vol. 30, no. 5, pp. 590–600, 2011.

[20] A. Prorok, M. A. Hsieh, and V. Kumar, "The Impact of Diversity on Optimal Control Policies for Heterogeneous Robot Swarms," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 346–358, 2017.

[21] T. H. Labella, M. Dorigo, and J.-L. Deneubourg, "Division of Labor in a Group of Robots Inspired by Ants' Foraging Behavior," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 1, no. 1, pp. 4–25, 2006.

[22] E. Castello, T. Yamamoto, Y. Nakamura, and H. Ishiguro, "Foraging Optimization in Swarm Robotic Systems Based on an Adaptive Response Threshold Model," *Advanced Robotics*, vol. 28, no. 20, pp. 1343–1356, 2014.

[23] H. Kurdi, J. How, and G. Bautista, "Bio-Inspired Algorithm for Task Allocation in Multi-UAV Search and Rescue Missions," in *AIAA Guidance, Navigation, and Control Conference*, San Diego, CA, USA, 2016.

[24] W. Liu, A. F. T. Winfield, J. Sa, J. Chen, and L. Dou, "Towards Energy Optimization: Emergent Task Allocation in a Swarm of Foraging Robots," *Adaptive Behavior*, vol. 15, no. 3, pp. 289–305, 2007.

[25] W. Liu and A. F. T. Winfield, "Modeling and Optimization of Adaptive Foraging in Swarm Robotic Systems," *The International Journal of Robotics Research*, vol. 29, no. 14, pp. 1743–1760, 2010.

[26] A. Martinoli, K. Easton, and W. Agassounon, "Modeling Swarm Robotic Systems: a Case Study in Collaborative Distributed Manipulation," *The International Journal of Robotics Research*, vol. 23, no. 4, pp. 415–436, 2004.

[27] K. Lerman, A. Martinoli, and A. Galstyan, "A Review of Probabilistic Macroscopic Models for Swarm Robotic Systems," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 143–152.

[28] N. Correll and A. Martinoli, "System Identification of Self-Organizing Robotic Swarms," in *Distributed Autonomous Robotic Systems 7*. Tokyo: Springer Japan, 2006, pp. 31–40.

[29] A. Prorok, N. Correll, and A. Martinoli, "Multi-level Spatial Modeling for Stochastic Distributed Robotic Systems," *The International Journal of Robotics Research*, vol. 30, no. 5, pp. 574–589, 2011.

[30] A. Kanakia, J. Klingner, and N. Correll, "A Response Threshold Sigmoid Function Model for Swarm Robot Collaboration," *Distributed Autonomous Robotic Systems*, pp. 193–206, 2016.

[31] D. Dimitrov and S. C. Sung, "Top responstiveness and Nash stability in coalition formation games," *Kybernetika*, vol. 42, no. 4, pp. 453–460, 2006.

[32] A. Darmann, E. Elkind, S. Kurz, J. Lang, J. Schauer, and G. Woeginger, "Group Activity Selection Problem," in *Proceedings of the 8th International Conference on Internet and Network Economics*, Liverpool, UK, 2012, pp. 156–169.

[33] A. Darmann, "Group Activity Selection from Ordinal Preferences," in *Algorithmic Decision Theory. ADT 2015. Lecture Notes in Computer Science*, ser. Lecture Notes in Computer Science, T. Walsh, Ed. Berlin, Heidelberg: Springer Cham, 2015, vol. 9346, pp. 35–51.

[34] E. Sahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 10–20.

[35] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable Self-assembly in a Thousand-robot Swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[36] S. C. Sung and D. Dimitrov, "On Myopic Stability Concepts for Hedonic Games," *Theory and Decision*, vol. 62, no. 1, pp. 31–45, 2007.

[37] M. Karakaya, "Hedonic Coalition Formation Games: A New Stability Notion," *Mathematical Social Sciences*, vol. 61, no. 3, pp. 157–165, 2011.

[38] H. Aziz and F. Brandl, "Existence of Stability in Hedonic Coalition Formation Games," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, Valencia, Spain, 2012, pp. 763–770.

[39] J. Guerrero and G. Oliver, "Multi-robot Coalition Formation in Real-time Scenarios," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1295–1307, 2012.

[40] O. Shehory and S. Kraus, "Feasible Formation of Coalitions Among Autonomous Agents in Nonsuperadditive Environments," *Computational Intelligence*, vol. 15, no. 3, pp. 218–251, 1999.

[41] C. Nam and D. A. Shell, "Assignment Algorithms for Modeling Resource Contention in Multirobot Task Allocation," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 889–900, 2015.

[42] L. B. Johnson, H.-L. Choi, and J. P. How, "The Role of Information Assumptions in Decentralized Task Allocation: A Tutorial," *IEEE Control Systems*, vol. 36, no. 4, pp. 45–58, 2016.

[43] W. Saad, Z. Han, M. Debbah, and A. Hjørungnes, "A Distributed Coalition Formation Framework for Fair User Cooperation in Wireless Networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 9, pp. 4580–4593, 2009.

[44] W. Saad, Z. Han, T. Basar, M. Debbah, and A. Hjørungnes, "Hedonic Coalition Formation for Distributed Task Allocation among Wireless Agents," *IEEE Transactions on Mobile Computing*, vol. 10, no. 9, pp. 1327–1344, 2011.

[45] R. E. Allen, A. A. Clark, J. A. Starek, and M. Pavone, "A Machine Learning Approach for Real-Time Reachability Analysis Ross," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA, 2014, pp. 2202–2208.

[46] M. Abdelkader, H. Jaleel, and J. S. Shamma, "A Distributed Framework for Real Time Path Planning in Practical Multi-agent Systems," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 10 626–10 631, 2017.

[47] M. Otte and N. Correll, "Dynamic teams of robots as ad hoc distributed computers: reducing the complexity of multi-robot motion planning via subspace selection," *Autonomous Robots*, 2018.

Chapter 4

A Comparative Study of Game-theoretical and Markov-chain-based Approaches to Division of Labour in a Robotic Swarm

4.1 Introduction

In this chapter, we are particularly interested in understanding the differences between two frameworks proposed in the previous chapters: the game-theoretical one [1] and the Markov-chain-based one [2]. We consider a mission scenario where a swarm of robots (or *agents*) are supposed to make multiple disjoint teams (or *coalitions*) responsible for individual tasks. The game-theoretical framework, called *GRAPE (GRoup Agent Partitioning and placing Event)*, models the agents as selfish players who attempt to make coalitions by unilaterally following their individual preferences regarding co-worker candidates. In the Markov-chain-based framework, to which we refer as *LICA-MC (Local Information Consistency Assumption-based Markov Chain framework)*, the agents stochastically behave using a row-stochastic vector of a time-inhomogeneous Markov

matrix, which is distributedly and adaptively generated based on local information feedback. In both approaches, each agent chooses its action independently without negotiating with the other agents, and thus, they can be classified as *self-organised* approaches [3]. Since self-organised approaches generally impose less communicational and computational burdens on agents when compared with other types of frameworks such as *orchestrated* ones, which need additional negotiation or voting mechanism to find a social agreement, they are considered more suitable for coordinating a large number of agents [4].

However, within this category, GRAPE and LICA-MC still have their own distinct features that provide different benefits and costs, which will be explored and compared in this study. For the comparison, we implement the frameworks into a labour division problem of a robotic swarm [5–7], the objective of which is to make a set of agents (i.e., workforce) distribute themselves autonomously into a set of tasks in proportion to the demands (i.e., workload) of the tasks. Primarily, we will focus on the required convergence time until achieving the desired collective behaviour, because this is mainly linked to real-time implementability of the frameworks. Furthermore, we discuss other implicit advantages of the frameworks such as mission suitability, additionally-built-in decision-making functions, and sensitivity to traffic congestion or robots' mobility.

This chapter is organised as follows. After briefly introducing the two frameworks in Section 4.2, we describe in Section 4.3 key components of this study such as the problem domain, how to implement the frameworks into the domain, and evaluation metrics. Then, Section 4.4 discusses the comparative benefits and costs of the frameworks, depending on various environmental factors. Finally, concluding remarks are followed in Section 4.5.

Definitions and Notations

$\mathcal{A} = \{a_1, a_2, ..., a_{n_a}\}$ denotes a set of n_a agents, and $\mathcal{T} = \{t_1, t_2, ..., t_{n_t}\}$ is a set of n_t tasks. $\mathbf{v} \in \mathbb{P}^n$ is a row-stochastic vector such that $\mathbf{v} \geq \mathbf{0}$ and $\mathbf{v} \cdot \mathbf{1}^\top = 1$, where $\mathbf{0}$ and $\mathbf{1}$ denote a row vector with each element being zero and that with one, respectively. $\mathbf{v}[i]$ indicates the i-th element of vector $\mathbf{v}$. $\lceil x \rceil$ indicates the ceiling function that maps x into the least integer that is greater than or equal to x.

Table 4.1: Nomenclature

Symbol	Description
$\mathcal{A}$	a set of n_a agents $\{a_1, a_2, ..., a_{n_a}\}$
$\mathcal{T}$	a set of n_t tasks $\{t_1, t_2, ..., t_{n_t}\}$
Θ	The desired swarm distribution
$\mathbf{x}_k$	The current swarm distribution at time step k
$\mathbf{n}_k[j]$	The current number of agents assigned to task t_j at time step k
D_H	Hellinger Distance (i.e., convergence error)
$\mathtt{t}_{k^\star}$	The convergence time until reaching the target value $D_H^\star$
$\mathbf{C}$	The path network of the tasks
$\mathbf{d}[j, l]$	The distance between task t_j and t_l based on the given path network
$\mathcal{N}_k(i)$	Agent a_i's neighbour agent set given the communication network at k
v	The moving speed of each robot
q	The road capacity (i.e., the number of lanes)
$\mathtt{t}_{col}$	The time separation for collision avoidance in a lane
d_G	The graph diameter of the agent communication network
Π^k	the *partition* at time step k that partitions the agent set $\mathcal{A}$, $\Pi^k = \{\mathcal{S}_1^k, \mathcal{S}_2^k, ..., \mathcal{S}_{n_t}^k\}$
$\mathcal{S}_j$	the task-specific robotic coalition for t_j
$\Pi(i)$	the index of the task to which agent a_i is assigned given Π
u_j^i	Agent a_i's individual utility (interest) regarding task t_j
M_k	The stochastic policy matrix at time step k

4.2 The Frameworks: GRAPE and LICA-MC

Given a swarm of robots $\mathcal{A}$ with a relatively fewer number of tasks $\mathcal{T}$ (i.e., $n_a > n_t$), one of the decision-making issues in multi-robot cooperation is to properly make disjoint (robotic) coalitions for the tasks.

GRAPE regards each robot as a self-interested agent in an *anonymous hedonic game* [8, 9], where the agent's interest on each task varies depending on the number of agents in the task (called *participants*). In this framework, a *partition* refers to a set $\Pi = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{n_t}\}$ that disjointly partitions the agent set $\mathcal{A}$, where $\mathcal{S}_j \subseteq \mathcal{A}$ is the coalition for task t_j. Roughly speaking, given the current partition at (algorithmic) time step k, denoted by Π^k, an arbitrary agent unilaterally selects the most preferred task according to its preference, and updates and broadcasts the new partition information (i.e., Π^{k+1}) to its neighbour agents. This process continues until when the entire agents converge towards a social agreement called *Nash stable partition*, in which every agent cannot be unilaterally better off so that they do not have any conflict (see footnote[1] for the formal description). It is worth noting that the entire agents do not need to physically move from a task to another task until finding a Nash stable partition, which is assumed throughout this chapter. In our previous work [1], we found that if each agent's interest on each task is decreasing with regard to the number of participants (we refer to this property as *SPAO (Single-Peaked-At-One)* [1, Definition 4]), a Nash stable partition can be always determined in a decentralised manner within polynomial-time algorithmic complexity, bounded by $O(d_G n_a^2)$, even under a strongly-connected communication network of the agents. Here, d_G is the graph diameter of the network.

Alternatively, in a Markov-chain-based approach, the dynamics of a a robotic swarm is considered as a continuum model governed by a Markov process: $\mathbf{x}_{k+1} = \mathbf{x}_k M_k$, where $\mathbf{x}_k \in \mathbb{P}^{n_t}$ denotes *the current swarm distribution* (i.e., population fraction) over the given n_t tasks, and $M_k \in \mathbb{P}^{n_t \times n_t}$ is the state transition matrix at time step k. At every time step k, each agent at task t_j computes its local *stochastic policy* $M_k[j, l]$ for $\forall t_l \in \mathcal{T}$,

[1] We quantify agent a_i's interest on task t_j by *individual utility* $u_j^i \in \mathbb{R}$, which is a function of the number of participants for the task, $p_j \in \{1, 2, ..., n_t\}$. A partition is said to be *Nash stable* if it holds for every agent $a_i \in \mathcal{A}$ that $u^i_{\Pi(i)} \geq u^i_j$, $\forall t_j \in \mathcal{T}$. Here, $\Pi(i)$ indicates the index of the task in which agent a_i is involved when Π is given.

where each element represents the probability that the agent will transition to task t_l before the next time step. According to the policy, the agent stochastically moves to another task (or stay in the current task). The entire agents can averagely converge to a desired status regardless of any initial condition if the Markov process is designed to be *ergodic* (i.e., the powers converge to a rank-one matrix) [10]. Using a time-homogeneous Markov process causes the trade-off between convergence rate and unnecessary transitions after reaching the desired status [11]. The trade-off was mitigated in the existing work [7], where a time-inhomogeneous Markov process using global information-based feedback was proposed in order that a robotic swarm can quickly converge in the initial phase and gradually settle down as approaching the desired collective state. Using biological inspiration, our previous work [2] recently proposed LICA-MC and showed that local information is enough to generate effective feedback as that in [7][2]. Moreover, it was shown in [2] that LICA-MC is more robust against asynchronous environments (e.g., unexpected communication disconnection between agents) compared with [7].

4.3 Study Formulation

This section describes key settings for this comparison study, such as the mission scenario, evaluation metrics, and how we implement the frameworks.

4.3.1 Mission Scenario: Swarm Distribution Guidance Problem

Labour division in a robotic swarm can be defined by *Swarm Distribution Guidance Problem* (SDGP) [5]: a swarm of robots $\mathcal{A}$ have to autonomously distribute themselves into a set of tasks $\mathcal{T}$, satisfying *the desired swarm distribution* Θ over the tasks. Here, the desired swarm distribution $\Theta \in \mathbb{P}^{n_t}$ is a row-stochastic vector such that each element

[2]Technically, our previous work [2] uses a function $f(\bar{\mathbf{x}}_k, \bar{\Theta})$ as the feedback to adaptively construct M_k, whereas $f(\mathbf{x}_k, \Theta)$ is utilised in [7]. Here, $\bar{\mathbf{x}}_k[j] = \mathbf{n}_k[j]/\sum_{s:\mathbf{C}[j,s]=1} \mathbf{n}_k[s]$ and $\bar{\Theta}[j] = \Theta[j]/\sum_{s:\mathbf{C}[j,s]=1} \Theta[s]$. Note that $\mathbf{n}_k$, Θ, and $\mathbf{C}$ will be defined in Section 4.3.1. In a nutshell, agents at task t_j in [7] need to communicate with the entire agents somehow (even in a multi-hop manner) to obtain $\mathbf{x}_k$, meanwhile those in [2] only have to do so for $\bar{\mathbf{x}}_k$ with other agents in all the neighbour tasks such that $\mathbf{C}[j, s] = 1$.

$\Theta[j] > 0$ indicates the desired population fraction (e.g., relative workforce demand) for task $t_j \in \mathcal{T}$.

As a performance index, we use *Hellinger Distance* (denoted by D_H), which measures the similarity between the current status $\mathbf{x}_k$ and the desired one Θ. This is defined as

$$D_H(\Theta, \mathbf{x}_k) := \frac{1}{\sqrt{2}} \sqrt{\sum_{\forall t_j \in \mathcal{T}} \left(\sqrt{\Theta[j]} - \sqrt{\mathbf{x}_k[j]} \right)^2}, \tag{4.1}$$

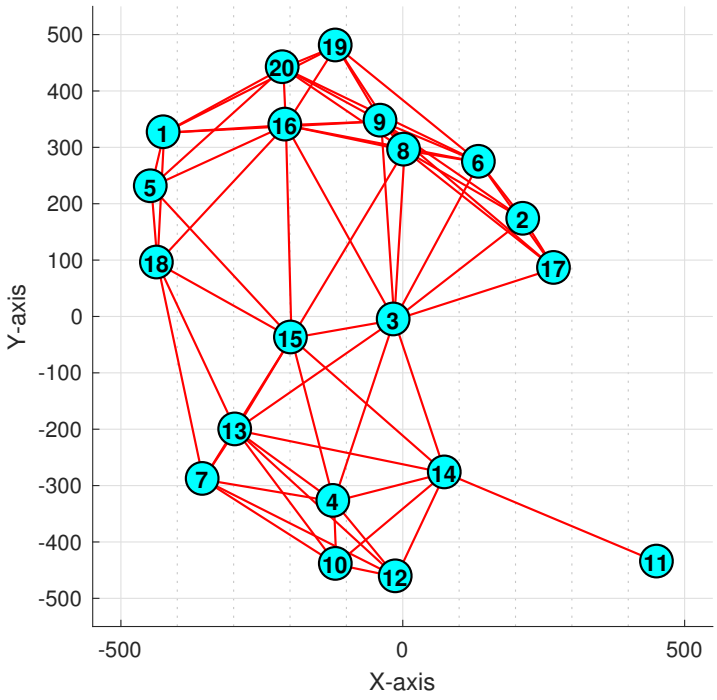
where, $\mathbf{x}_k[j] = \mathbf{n}_k[j]/n_a$, and $\mathbf{n}_k[j]$ is the current number of agents assigned to (not necessarily located at) task t_j at time step k. For GRAPE, $\mathbf{n}_k[j]$ is equivalent to $|\mathcal{S}_j^k|$. Note that this chapter interchangeably uses the term *convergence error* to refer to Hellinger Distance.

For the comparison, we will investigate the actual necessary time $\mathtt{t}_{k^\star}$ (called *convergence time* in this chapter) for the agents not only to be assigned but also to "physically converge" towards a distribution such that $D_H(\Theta, \mathbf{x}_{k^\star}) \leq D_H^\star$, where $D_H^\star$ is the target level of convergence error. Here, $\mathtt{t}_{k^\star} \in \mathbb{R}$ is a function that maps $k^\star$ of time steps occurred in an algorithm onto the actual time it took. This function may be different for different algorithms. Its details will be explained in Section 4.3.2.

Given the tasks' minimum spanning tree (MST) based on their spatial closeness, we assume that any two tasks have a direct path between them if their distance is shorter than the tree's maximum-length edge. The direct path availabilities over the tasks (or called *the path network*) can be represented by the matrix $\mathbf{C} \in \{0,1\}^{n_t \times n_t}$, in which $\mathbf{C}[j,l] = 1$ means that t_j and t_l have the direct path between them. Note that all its diagonal entries are set to be one, and $\mathbf{C}$ is assumed to be symmetric. Since the path network are at least strongly-connected, agents can somehow move from a task to another. Let $\mathbf{d}[j,l]$ denote the shortest path distance between task t_j and t_l based on the path network.

For both frameworks, we equally assume that each agent in task t_j can directly communicate with any agent in t_l if $\mathbf{C}[j,l] = 1$. Given the agent communication network for time step k, we denote by $\mathcal{N}_k(i)$ the set of agent a_i's *neighbour agents* (i.e., whom it can directly communicate). A scenario example is depicted in Figure 4.1.

Since this study focuses on the performance comparison in high-level mission planning and task allocation, we assume that each agent is capable of knowing locations



(a)

i	$\Theta[i]$
1	2 %
2	15 %
3	5 %
4	10 %
5	3 %
6	3 %
7	1 %
...	...
19	2 %
20	3 %

s.t. $\sum_{\forall t_j} \Theta[j] = 1$

(b)

Figure 4.1: A mission scenario example with $n_t = 20$ of tasks: (a) the tasks' positions (indicated by circles) and their physical connections, i.e., available paths (represented by red lines); (b) the desired swarm distribution over the tasks

of itself and all the given tasks, and able to do collision avoidance behaviours against the other agents. It is also assumed that every agent has the mission description information, such as Θ, $D_H^\star$, $\mathbf{C}$, and $\mathbf{d}$, as well as can access to local information of its neighbour agents in $\mathcal{N}_k(i)$.

4.3.2 Evaluation Metrics

A framework's convergence time $\mathtt{t}_{k^\star}$ in general consists of the time spent by agents on communication, computation, and physical transition. However, strategical distinctions of GRAPE and LICA-MC result in different forms of convergence time with regard to algorithmic time steps. In GRAPE, making an agreed task assignment, which needs

agents' recursive computational and communicational behaviours only, precedes the physical movement to the assigned tasks. On the contrary, agents in LICA-MC compute, communicate, and move to a task at every algorithmic step. Let $\mathtt{t}^G_{k^\star}$ and $\mathtt{t}^M_{k^\star}$ denote the convergence time for GRAPE and LICA-MC, respectively, and they can be defined as follows:

$$\mathtt{t}^G_{k^\star} := \sum_{k=0}^{k^\star-1} \Big(\Delta\mathtt{t}^G_{comm}(k) + \Delta\mathtt{t}^G_{comp}(k) \Big) + \Delta\mathtt{t}^G_{trans}, \tag{4.2}$$

$$\mathtt{t}^M_{k^\star} := \sum_{k=0}^{k^\star-1} \Big(\Delta\mathtt{t}^M_{comm}(k) + \Delta\mathtt{t}^M_{comp}(k) + \Delta\mathtt{t}^M_{trans}(k) \Big), \tag{4.3}$$

where $\Delta\mathtt{t}^G_{comm}(k)$ or $\Delta\mathtt{t}^M_{comm}(k)$ is the time scale for the entire agents to communicate and collect necessary local information for executing the corresponding algorithm for time step k; and $\Delta\mathtt{t}^G_{comp}(k)$ or $\Delta\mathtt{t}^M_{comp}(k)$ is the time spent to compute local decision rationales for time step k. For LICA-MC, $\Delta\mathtt{t}^M_{trans}(k)$ indicates the necessary time for the agents physically move from one task to one another for time step k. For GRAPE, $\Delta\mathtt{t}^G_{trans}$ is the total transition time of the entire agents from their initial states to the final assigned tasks.

The communication time, i.e., $\Delta\mathtt{t}^G_{comm}(k)$ or $\Delta\mathtt{t}^M_{comm}(k)$, varies depending on the current communication network amongst the agents. In GRAPE, the agent network is consistent during the decision-making process for finding a Nash stable partition. Assuming synchronisation (i.e., for each time step, every agent waits for all the other agents to collect necessary local information), we regard that $\Delta\mathtt{t}^G_{comm}(k) = \mathtt{t}_{uc} \cdot \max_{\forall a_i}\{|\mathcal{N}^G(i)|\}$, where $\mathtt{t}_{uc}$ is the unit communication time between two agents, and $\mathcal{N}^G(i)$ is agent a_i's neighbour agent set in GRAPE. On the contrary, agents' communication network in LICA-MC changes as they move around for each time step, and thus $\Delta\mathtt{t}^M_{comm}(k) = \mathtt{t}_{uc} \cdot \max_{\forall a_i}\{|\mathcal{N}^M_k(i)|\}$. Experimentally, we observed that the time average value of $\max_{\forall a_i}\{|\mathcal{N}^M_k(i)|\}$ in LICA-MC and the value of $\max_{\forall a_i}\{|\mathcal{N}^G(i)|\}$ in GRAPE are almost the same. Hence, we consider that

$$\Delta\mathtt{t}_{comm} := \Delta\mathtt{t}^G_{comm}(k) \approx \Delta\mathtt{t}^M_{comm}(k). \tag{4.4}$$

Regarding the computation time (i.e., $\Delta\mathtt{t}^G_{comp}(k)$ and $\Delta\mathtt{t}^M_{comp}(k)$), since each agent in either of the frameworks investigates n_t tasks at each time step, we assume that

$$\Delta\mathtt{t}_{comp} := \Delta\mathtt{t}^G_{comp}(k) \approx \Delta\mathtt{t}^M_{comp}(k). \tag{4.5}$$

For the transition time (i.e., $\Delta \mathtt{t}^{M}_{trans}(k)$ and $\Delta \mathtt{t}^{G}_{trans}$), we assume a simple congestion model as follows. Every agent's moving speed v is constant, and all the available paths between the tasks have the same *road capacity* q (e.g., the number of (one-way) lanes). Let $\mathtt{t}_{col}$ denote the *time separation* for collision avoidance between any two consecutively moving robots in a lane. When a large number of robots (more than q) have to move through a path, at most q of the robots can begin to pass the path the time interval $\mathtt{t}_{col}$ after the previous robot group left.

For LICA-MC, let $\mathbf{n}^{trans}_{k}[j,l]$ denote the number of transitioning agents from t_j to t_l for time step k. Then, the corresponding transition time can be defined as

$$\Delta \bar{\mathtt{t}}^{M}_{trans}[j,l](k) := \frac{\mathbf{d}[j,l]}{v} + \mathtt{t}_{col} \cdot \Big(\Big\lceil \frac{\mathbf{n}^{trans}_{k}[j,l]}{q} \Big\rceil - 1 \Big). \tag{4.6}$$

Here, the first term $\mathbf{d}[j,l]/v$ means the baseline transition time of a robot. The second term indicates additional required time if the number of moving agents is more than q, assuming that the robots marginally satisfy the safety separation time $\mathtt{t}_{col}$. We regard the entire agents' transition time for each time step k as

$$\Delta \mathtt{t}^{M}_{trans}(k) := \max_{\forall j \neq l} \{ \Delta \bar{\mathtt{t}}^{M}_{trans}[j,l](k) \}. \tag{4.7}$$

On the contrary, in GRAPE, the transition time is not related to algorithmic steps because all the agents start to move after finding a social agreement. Let $\mathbf{d}^{\star}_{max} := \max_{\forall a_i} \big\{ \mathbf{d}\big[\Pi^0(i), \Pi^{k^\star}(i)\big] \big\}$ denote the longest path amongst those every agent $a_i \in \mathcal{A}$ has to move from its initial status $t_{\Pi^0(i)}$ to the finally assigned task $t_{\Pi^{k^\star}(i)}$. Conservatively, we regard the transition time $\Delta \mathtt{t}^{G}_{trans}$ in Equation (4.2) as the time for all the agents to traverse the longest path simultaneously:

$$\Delta \mathtt{t}^{G}_{trans} := \frac{\mathbf{d}^{\star}_{max}}{v} + \mathtt{t}_{col} \cdot \Big(\Big\lceil \frac{n_a}{q} \Big\rceil - 1 \Big) \tag{4.8}$$

In summary, Equations (4.2) and (4.3) can be rewritten as

$$\mathtt{t}^{G}_{k^\star} = (\Delta \mathtt{t}_{comm} + \Delta \mathtt{t}_{comp}) \cdot k^\star + \Delta \mathtt{t}^{G}_{trans}, \tag{4.9}$$

$$\mathtt{t}^{M}_{k^\star} = (\Delta \mathtt{t}_{comm} + \Delta \mathtt{t}_{comp}) \cdot k^\star + \sum_{k=0}^{k^\star - 1} \Delta \mathtt{t}^{M}_{trans}(k), \tag{4.10}$$

where $\Delta\mathtt{t}^{G}_{trans}$ and $\Delta\mathtt{t}^{M}_{trans}(k)$ are from Equations (4.8) and (4.7), respectively. Although they are not precise models, we believe that it is enough to use them to explore each framework's relative benefits in this comparative study.

4.3.3 Implementations

4.3.3.1 GRAPE

For implementation of GRAPE into a decision-making scenario, agents' individual utilities (please refer to footnote 1 of Section 4.2 for the definition) are the main components that we need to carefully design in order for the desired collective behaviour to emerge. In this study, we define agent a_i's individual utility with regard to task t_j as

$$u^i_j := r_j/p_j - w_c \cdot c^i_j, \tag{4.11}$$

where r_j is task t_j's constant reward which will be equally shared by p_j of co-workers; c^i_j is the agent's individual cost for executing the task; and $w_c \in \mathbb{R}^+$ is the weight factor for the cost relative to the shared reward. We regard that the agent can be rewarded as much as it contributes to the task's work demand. Thus, the task's reward is set to be proportional to its demand as follows:

$$r_j := n_a \cdot \Theta[j]. \tag{4.12}$$

Since the evaluation metric shown in Section 4.3.2 (i.e., convergence time $\mathtt{t}_{k^\star}$) is affected by transitioning time, it is desirable to reduce unnecessary travel distance of agents. Hence, we set the cost c^i_j in Equation (4.11) as a function of the distance from the agent's initial status to task t_j:

$$c^i_j := \mathbf{d}[\Pi^0(i), j]/\mathbf{d}_{max}, \tag{4.13}$$

where $\mathbf{d}_{max} := \max_{\forall j \forall l} \mathbf{d}[j, l]$ is the maximum value amongst the distances between any two tasks.

To examine the effect of the weight factor w_c, we conduct 100 runs of Monte-Carlo simulations with various $w_c \in \{100, 10, 1, 0.1, 0.01\}$. We randomly generate each instance of SDGP with $n_t = 20$ and $n_a = 400$ within a 500×500 environment, and

then investigate how well the resultant Nash stable partition satisfies with the desired distribution (i.e., convergence error), the number of required algorithmic time steps, and the total travelling costs of the agents. The statistical results in Figure 4.2 show that there is a trade-off of convergence error versus the others. As w_c increases, the agents become more reluctant to move to any tasks even if they could be provided a higher reward than the current one. Consequently, the number of time steps happened, the number of transitioning agents, and the resultant travelling costs become lower, as presented in Figure 4.2(b), (c), and (d), respectively. On the contrary, with a lower value of w_c, the agents tend to be more concerned with obtaining the shared rewards and thus result in lower convergence error, as shown in Figure 4.2(a). A possible reason why the required number of time steps have declined at $w_c = 0.01$, as shown in Figure 4.2(b), is that the agents became almost indifferent to the travelling costs, which makes their individual preferences simpler and reduces the algorithmic iterations occurred. For the rest of this chapter, we will use $w_c = 0.1$ since it yields reasonable performances in terms of convergence error, iterations, and travelling costs.

4.3.3.2 LICA-MC

We use one of the implementation examples in our previous work [2, Section VI-A], which is designed to address SDGP with minimising travelling costs. The followings are its essential definitions, so please refer to the paper for more details.

The stochastic policy for each agent in task t_j to move towards task t_l is such that

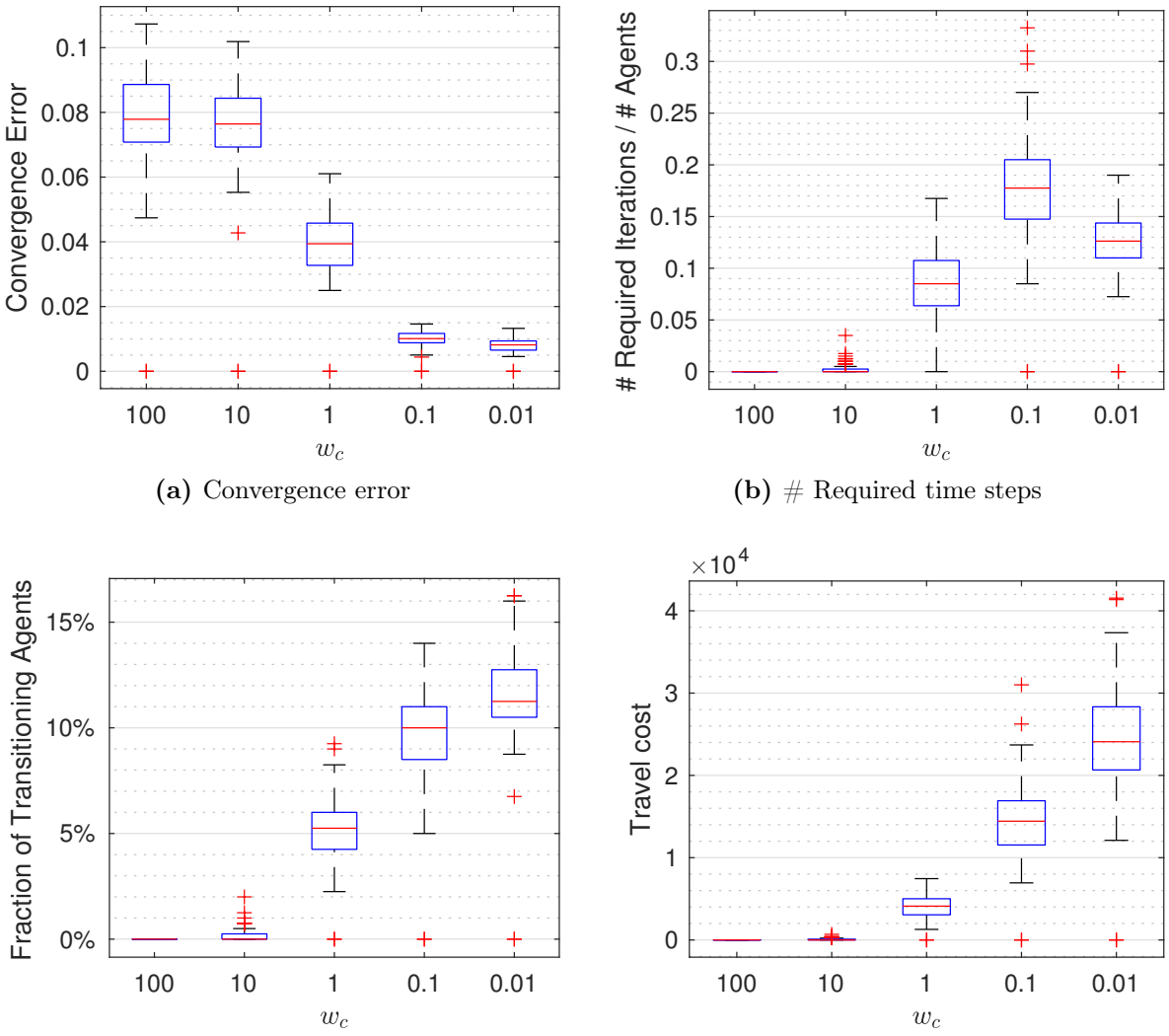
$$M_k[j,l] := (1 - \omega_k[j])P_k[j,l] + \omega_k[j]\mathcal{S}_k[j,l]. \tag{4.14}$$

The primary policy matrix P_k is defined by, for its off-diagonal elements (i.e., $\forall l \neq j$),

$$P_k[j,l] := \begin{cases} \epsilon_\Theta \Theta[l] \max(\bar{\xi}_k[j], \bar{\xi}_k[l])(1 - \frac{\mathbf{d}[j,l]}{\mathbf{d}_{max}+\epsilon_E}) & \text{if } \mathbf{C}[j,l] = 1 \\ 0 & \text{otherwise,} \end{cases} \tag{4.15}$$

and for its diagonal elements,

$$P_k[j,j] := 1 - \sum_{\forall l \neq j} P_k[j,l], \tag{4.16}$$



(a) Convergence error

(b) # Required time steps

(c) Fraction of transitioning agents

(d) Total travel cost

Figure 4.2: Effectiveness of the weight factor w_c in GRAPE

where ϵ_Θ is the factor to enhance convergence rate [2, Remark 4]

$$\epsilon_\Theta := \min\{\frac{1}{\sum_{\forall s\neq j:\mathbf{C}[j,s]=1}\Theta[s]}, \frac{1}{\sum_{\forall s\neq l:\mathbf{C}[l,s]=1}\Theta[s]}\}; \tag{4.17}$$

$\bar{\xi}_k[j]$ is the local feedback gain

$$\bar{\xi}_k[j] := \big(\kappa \cdot \frac{|\bar{\Theta}[j] - \bar{\mathbf{x}}_k[j]|}{\bar{\Theta}[j]}\big)^\alpha \tag{4.18}$$

being saturated to $[\epsilon_\xi, 1]$ if the value lies outside this range; and ϵ_E, κ, α, and ϵ_ξ are design parameters. Note that $\bar{\Theta}$ and $\bar{\mathbf{x}}_k$ are defined in footnote 2 of Section 4.2. *The secondary policy matrix* $\mathcal{S}_k$ is set to be the $n_t \times n_t$ identity matrix. The weight factor between $P_k[j,l]$ and $\mathcal{S}_k[j,l]$ is determined by $\omega_k[j] \in [0,1)$, which is defined as

$$\omega_k[j] := \exp(-\lambda k) \cdot \frac{\exp(\beta(\bar{\Theta}[j] - \bar{\mathbf{x}}_k[j]))}{\exp(\beta|\bar{\Theta}[j] - \bar{\mathbf{x}}_k[j]|)}, \tag{4.19}$$

where λ and β are design parameters. For this study, all the design parameters are set as follows: $\epsilon_E = 0.1$ for (4.15); $\kappa = 0.1$, $\alpha = 0.65$, and $\epsilon_\xi = 10^{-9}$ for (4.18); and $\lambda = 10^{-6}$ and $\beta = 1.8 \times 10^5$ for (4.19).

4.4 Comparative Results and Discussion

4.4.1 Numerical Experiments

This section examines the effect of the number of agents on each framework's performances. We randomly generate 100 scenarios of SDGP with $n_t = 20$ of tasks in a 500×500 environment. For each scenario, we implement GRAPE and LICA-MC with $n_a \in \{200, 400, 800, 1600, 3200\}$ of agents. The initial and desired swarm distributions, which are also randomly generated, remain consistent across the same scenario regardless of different n_a. Note that we will also include the LICA-MC results with infinite agents. Such results can be simply obtained by assuming that the agents exactly follow the Markov process due to *the law of large numbers*.

Figure 4.3(a) shows the comparative result for a particular scenario of the experiments. The dotted lines indicate the converging behaviours of agents in GRAPE, while

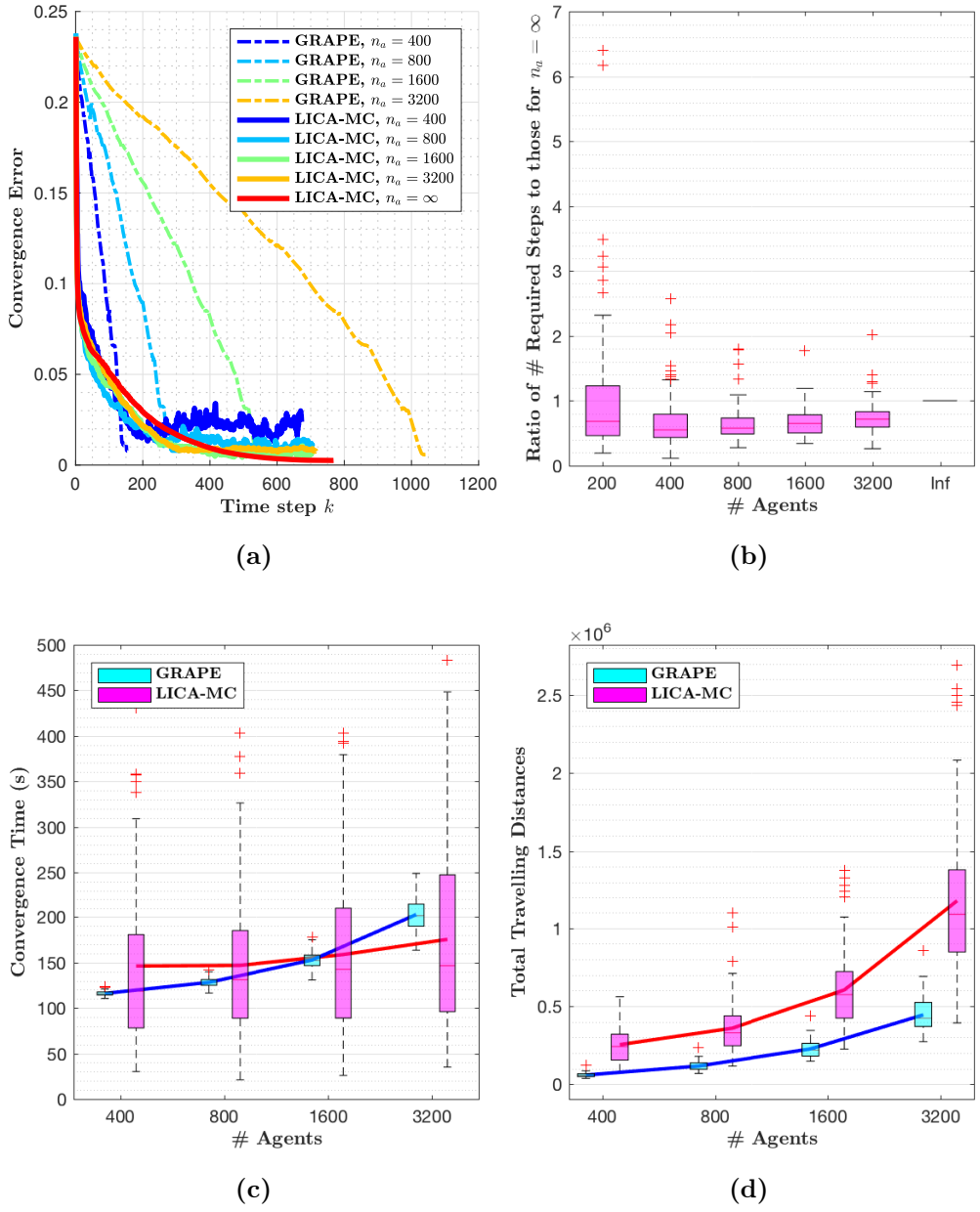


Figure 4.3: Performance Comparison between GRAPE and LICA-MC: (a) the converging behaviours of agents (for a specific scenario); (b) the number of time steps for scenarios with various n_a to those with $n_a = \infty$ in LICA-MC; (c) the convergence time (i.e., $\mathtt{t}_{k^\star}$); and (d) the total travelling distance of agents until achieving $D_H^\star = 0.03$.

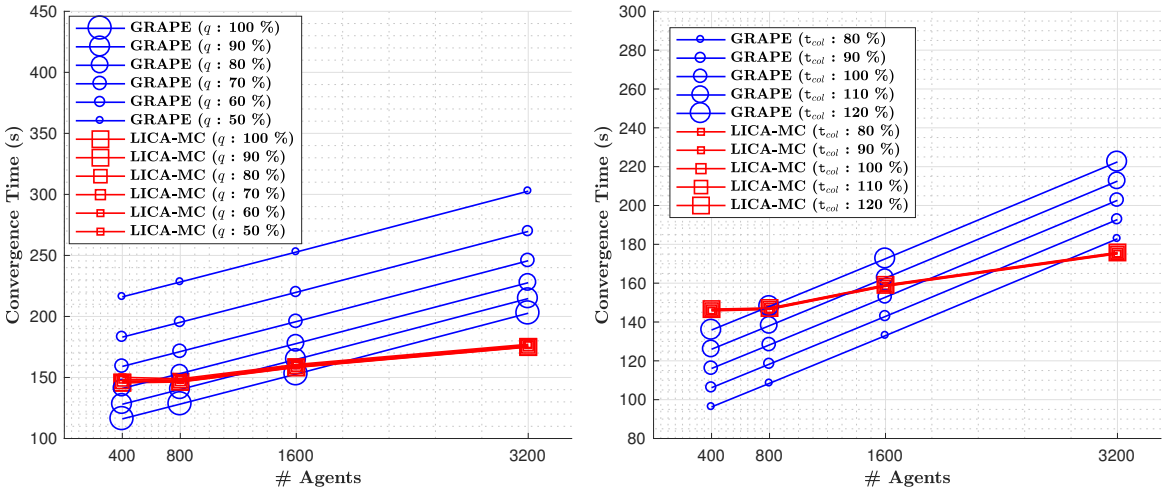
the solid lines are those in LICA-MC. It is presented that even though n_a increases in LICA-MC, the collective behaviours almost remain unchanged. Precisely, they become close to the result of the case of $n_a = \infty$. This is also confirmed by Figure 4.3(b) showing that the difference of the required time steps for the case with a certain n_a from those for infinite agents becomes smaller as n_a grows. Moreover, Figure 4.3(a) indicates that the residual error becomes lower with a larger number of agents, as pointed out in [7, Theorem 6]. On the contrary, GRAPE needs more iterations as n_a increases. Namely, LICA-MC is completely scalable with regard to n_a, while GRAPE has polynomial-time complexity.

Using the convergence time models in Equations (4.9) and (4.10), we evaluate the transition time $\mathtt{t}_{k^\star}$ of each framework to reach the convergence error level $D_H^\star = 0.03$. It is assumed that $\Delta\mathtt{t}_{comm} = 0.1$ sec and $\Delta\mathtt{t}_{comp} = 0.01$ sec. The robot speed v is set as the communication range per second, which implies that, for the equal distance, the order of transition time is approximately ten-times of that for communication. It is set that the road capacity is $q = n_a \cdot 0.01$ (i.e., 1% of the entire agents can begin to move at the same time), and the time separation is $\mathtt{t}_{col} = 1$ sec. The results are shown in Figure 4.3(c). As expected, the running time of GRAPE with regard to n_a is higher than that of LICA-MC. In this experiment, LICA-MC averagely gives better performance when n_a is greater than around 1500. Unlike the results in Figure 4.3(a), the convergence time resulted from LICA-MC gradually increases. This is because the transition time $\Delta\mathtt{t}^M_{trans}(k)$ takes more time as n_a increases given the same road capacity.

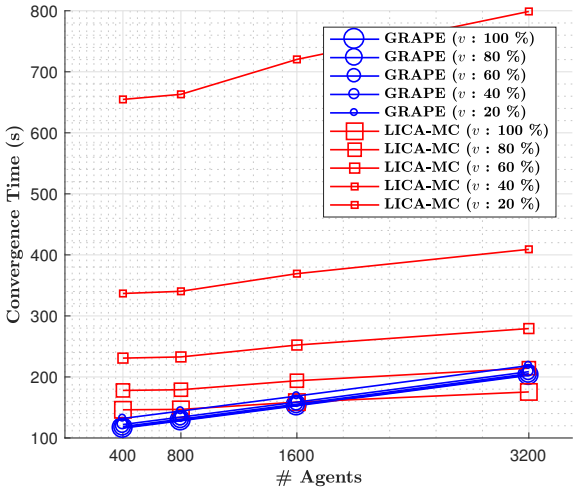
On the other hand, despite such randomly-generated scenarios, GRAPE gives similar converging performance within the same number of agents. Meanwhile, LICA-MC yields wide range of results, being affected by the randomness of the path network and the initial/desired swarm distributions.

Figure 4.3(d) indicates that GRAPE causes less travelling distances of agents than LICA-MC since, as described in Section 4.3.2, the agents in LICA-MC have to move around during the decision-making process.

Since some of the values set above might have different levels in reality, we conduct sensitivity analysis to explore the effects of the parameters such as the road capacity q, the time separation $\mathtt{t}_{col}$, and the robot speed v. Figures 4.4(a) and (b) show that the road capacity and the time separation have an influence on GRAPE, but not on



(a) Road capacity q **(b)** Time separation $\mathtt{t}_{col}$



(c) Robot speed v

Figure 4.4: Sensitivity analysis for the parameters such as q, $\mathtt{t}_{col}$, and v.

LICA-MC. It stems from the fact that all the agents under GRAPE have to move to the assigned tasks simultaneously after determining a social agreement. This makes GRAPE sensitive to traffic congestion. Hence, reducing q or increasing $\mathtt{t}_{col}$ causes higher convergence time in GRAPE. For LICA-MC, on the contrary, the figures imply that travelling agents per time step are well managed so that such severe traffic conditions do not affect the framework.

Figure 4.4(c) presents that LICA-MC is easily affected by the robot speed: as the robot speed reduces, LICA-MC needs more convergence time. This is because the reduced speed directly influences the transition time per algorithmic step $\Delta\mathtt{t}^{M}_{trans}(k)$, and this effect is amplified by the number of iterations happened, as shown in Equation (4.10). However, the parametric change of robot speed v hardly has an impact on GRAPE. Practically, it is highly possible that the actual robot speed is lower than that in the setting for Figure 4.3(c). This implies that GRAPE could provide faster convergence time even for $n_a > 1500$ of agents to some extent, depending on the robot speed.

4.4.2 Additional Discussions

GRAPE and LICA-MC have different strategic attributes. GRAPE is characterised by making a global agreed plan before action, whereas agents in LICA-MC myopically generate and follow their local policies without confirming any social agreement. From this difference, as noticed in Section 4.4.1, LICA-MC has greater scalability but may have longer transition time as well as unnecessary travelling costs. Thus, using LICA-MC may require a real-time battery recharging system for a swarm robots such as the wireless system in [12, 13]. We believe that GRAPE is more suitable for a scenario where robots do not need to move around during the mission, e.g., the cooperative jamming mission in [14]. However, continuous movement of agents in LICA-MC might be more preferred for some cases, for military examples, where the agents have to avoid attacks or where they are required to deceive foes. If we forcibly use LICA-MC without such transitioning, then the result would be just the same as that by the simple random decision-making mechanism (i.e. the roulette rule) in [15], which is neither optimised in terms of travelling costs nor have fast adaptability to dynamic environments (i.e., for

compensating partial loss of agents, it needs the entire decision-making process from the beginning again without exploiting existing information).

Agents in LICA-MC are only able to have *absolute* preferences over tasks (e.g., if $\Theta[j]$ is the highest value of all, task t_j is the most preferred by any agents). Thus, the preferences should reflect the globally desired status in view of a single operator. Contrarily, agents in GRAPE can have different *individual* preferences (e.g., some agents prefer task t_j but the others more benefit from task t_l even though the tasks are identically distant from them). This feature may provide benefits when different swarms from different organisations need to be accommodated. It is worth noting that, for the comparison in the previous section, we deliberately restricted the agents in GRAPE to having absolute preferences as those in LICA-MC.

The two frameworks have another inherent decision-making function in addition to that for task allocation. As shown in Section 4.4.1, LICA-MC includes a path planning function, but which should be separately addressed when GRAPE is utilised. On the other hand, when using LICA-MC, human operators have to determine what is the desired collective status in advance, and then provide this information to agents. This is not necessary for GRAPE because, as long as agents are given information about tasks, they can find a social agreement based on their individual preferences.

We can say that the two frameworks are operable based on local information, but technically there exist differences. It is desirable for agents in LICA-MC to collect information from all the others in their neighbour tasks for each time step. Possible asynchronisation, for which it is still assumed information collection from those in at least one neighbour task, may gracefully degrade the framework's performance (i.e., unnecessary travelling costs increase) [2]. Whereas, in GRAPE, each agent's decision-making loop can proceed even when the local information from a single neighbour agent is only available. A deficient communication network of agents may increase the convergence time but the suboptimality of the outcome almost remains the same [1].

We summarise some remarkable distinctions between GRAPE and LICA-MC into Table 4.2.

Table 4.2: Features Comparison between GRAPE and LICA-MC

	Game-theoretical (GRAPE)	Markov-chain-based (LICA-MC)
A	Individual utility	Local stochastic policy
B	SPAO [1, Definition 4]	Ergodicity [2, Definition 13]
C	$O(n_a^2)$	$O(1)$
D	One of neighbour agents	All agents within neighbour tasks
E	Make an agreement then act	Locally plan then act, and repeat
F	Individual preferences	Absolute preferences
G	Desired status determination	Path planning

A: The decision rationale for each agent

B: The condition guaranteeing a desired collective behaviour

C: Scalability with regard to the number of agents
(the required algorithmic time steps for convergence)

D: Other agents required to communicate with, for each time step
(at least a strongly-connected network is assumed)

E: The behavioural sequence of each agent

F: Applicable agent preferences

G: Additional built-in function except for task allocation

4.5 Conclusion

In this study, we have compared the pros and cons of the game-theoretical framework, GRAPE, and the Markov-chain-based framework, LICA-MC. For the comparison, we implemented both frameworks into a problem of robotic swarm labour division (i.e., swarm distribution guidance problem), and then introduced evaluation metric models in terms of their convergence performances. The numerical experiments showed that LICA-MC is more scalable, whereas GRAPE yields faster convergence time for a moderate number of agents and causes less total travelling costs. We also presented through the sensitivity analysis that possible traffic congestion may affect the performance of GRAPE, which is not the case for LICA-MC owing to its built-in path planning function. Meanwhile, it was shown that LICA-MC is sensitive to slower robot speed because of agents' behavioural sequence in the framework. Moreover, we discussed about the frameworks' additional features in terms of strategic advantages depending on missions, information-sharing requirements, and the ability to accommodate different interests of agents.

For a future work, we would like to perform a comparative study based on real-robot experiments.

References

[1] I. Jang, H.-S. Shin, and A. Tsourdos, "Anonymous Hedonic Game for Task Allocation in a Large-scale Multiple Agent System," *IEEE Transactions on Robotics (in press)*, 2018.

[2] I. Jang, H.-S. Shin, and A. Tsourdos, "Bio-Inspired Local Information-Based Control for Probabilistic Swarm Distribution Guidance," *arXiv:1711.06869 [cs.MA]*, 2017.

[3] A. Brutschy, G. Pini, C. Pinciroli, M. Birattari, and M. Dorigo, "Self-organized Task Allocation to Sequentially Interdependent Tasks in Swarm Robotics," *Autonomous Agents and Multi-Agent Systems*, vol. 28, no. 1, pp. 101–125, 2014.

[4] N. Kalra and A. Martinoli, "A Comparative Study of Market-Based and Threshold-Based Task Allocation," in *Distributed Autonomous Robotic Systems 7*, Tokyo, Ed. Springer Japan, 2006, pp. 91–101.

[5] B. Acikmese and D. S. Bayard, "A Markov Chain Approach to Probabilistic Swarm Guidance," in *American Control Conference*, Montreal, Canada, 2012, pp. 6300–6307.

[6] P. Zahadat and T. Schmickl, "Division of labor in a swarm of autonomous underwater robots by improved partitioning social inhibition," *Adaptive Behavior*, vol. 24, no. 2, pp. 1–11, 2016.

[7] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1103–1123, 2017.

[8] A. Darmann, E. Elkind, S. Kurz, J. Lang, J. Schauer, and G. Woeginger, "Group Activity Selection Problem," in *Proceedings of the 8th International Conference on Internet and Network Economics*, Liverpool, UK, 2012, pp. 156–169.

[9] A. Darmann, "Group Activity Selection from Ordinal Preferences," in *Algorithmic Decision Theory. ADT 2015. Lecture Notes in Computer Science*, ser. Lecture

Notes in Computer Science, T. Walsh, Ed. Berlin, Heidelberg: Springer Cham, 2015, vol. 9346, pp. 35–51.

[10] I. Ipsen, "Coefficients of Ergodicity: An Introduction," in *Numerical Analysis Seminar, NC State University (April 23, 2008)*, 2008.

[11] S. Berman, A. Halasz, M. A. Hsieh, and V. Kumar, "Optimized Stochastic Policies for Task Allocation in Swarms of Robots," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 927–937, 2009.

[12] F. Arvin, S. Watson, A. E. Turgut, J. Espinosa, T. Krajník, and B. Lennox, "Perpetual Robot Swarm: Long-Term Autonomy of Mobile Robots Using On-the-fly Inductive Charging," *Journal of Intelligent and Robotic Systems: Theory and Applications*, pp. 1–18, 2017.

[13] A. B. Junaid, Y. Lee, and Y. Kim, "Design and implementation of autonomous wireless charging station for rotary-wing UAVs," *Aerospace Science and Technology*, vol. 54, pp. 253–266, 2016.

[14] I. Jang, J. Jeong, H.-S. Shin, S. Kim, A. Tsourdos, and J. Suk, "Cooperative Control for a Flight Array of UAVs and an Application in Radar Jamming," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8011–8018, 2017.

[15] A. Jevtić, Á. Gutierrez, D. Andina, and M. Jamshidi, "Distributed Bees Algorithm for Task Allocation in Swarm of Robots," *IEEE Systems Journal*, vol. 6, no. 2, pp. 296–304, 2012.

Chapter 5

A Game-theoretical Approach to Heterogeneous Multi-Robot Task Assignment Problem with Minimum Workload Requirements

5.1 Introduction

Cooperation of a huge number of small-sized autonomous aerial robots, called *UAV swarms*, will play a major role in complex missions that existing operational concepts using a few large UAVs could not deal with [1]. Even if each robot (or called agent) is incapable of accomplishing a task alone, their cooperation will lead to successful outcomes because of the swarm system's robustness and adaptiveness. The possible applications include environmental monitoring [2], ad-hoc network relay [3], cooperative military missions [4], to name a few.

One of the main technical challenges for utilisation of a swarm is collective decision-making such as *multi-robot task assignment (MRTA) problem* [5]. Particularly, this chapter addresses the case where each task can not be fulfilled by a single agent but requires multiple agents to be completed, i.e., the task's minimum workload requirement should be met by cooperation of the agents. This case falls into ST-MR (Single-Task-

executable robot and Multi-Robot-required task) category [5]. Moreover, we take into account heterogeneous tasks and agents: each task needs a different type of workload (e.g., task t_1 requires sensing ability, whereas task t_2 demands transportation capability); and each agent also, depending on the tasks, has different work capacities (or efficiencies) and costs. The objective is to find an assignment, in a distributed manner, that satisfies the requirements of all the tasks while minimising the aggregated cost of assigned agents. We formulate this problem as *the minimisation version of the generalised assignment problem with minimum requirements (MinGAP-MR)*, which is defined in Section 5.2.

The (standard) generalised assignment problem (GAP) has been extensively studied over several decades. A general overview of GAP is available in [6] along with its real-life applications such as scheduling, transporting, and facility location. One of its key differences from MinGAP-MR is that GAP has maximum-capacity constraints for knapsacks instead of minimum requirements. In multi-robot system domain, a single robot has been typically considered as capable of executing multiple tasks (i.e., MT-SR (Multi-Task-executable robot and Single-Robot-required task) case [7,8]) within its work capacity limitation. Hence, some existing studies [9–11] address this type of MRTA problems by modelling as GAP or its variants, where each robot and task are regarded as a knapsack with its maximum-available size and an item to insert, respectively.

On the contrary, few works in the literature consider minimum-requirement constraints for knapsacks. Even for a single knapsack problem, approximation algorithms based on greedy heuristics are relatively recently proposed in [12, 13]. For multiple-knapsack cases, the works in [14, 15] study assignment problems of students to lectures where there is the minimum number of participants required for each lecture to launch. However, this constraint is only concerned with the cardinality of assigned agents, and thereby it is not suitable for heterogeneous agents. Although the work in [16] considers such agents along with knapsack's minimum requirements as well as maximum limitations, the suggested approximation algorithm does not always provide a feasible solution (i.e., the resultant assignment often violates the knapsack constraints).

This chapter proposes a game-theoretical approach for MinGAP-MR. As inspired by *coalitional games* [17, Chap 7], we regard each robot as a selfish player (i.e., item) who wants to join a task-specific coalition (i.e., knapsack) that minimises its cost. The

objective of this game is to find a *Nash stable partition*, i.e., a set of disjoint coalitions in which no agent will unilaterally deviate. To avoid possible conflicts between the players, we adopt a self-learning scheme by which every player gradually penalises its previously chosen coalition. In this chapter, we prove that the proposed approach always determines a Nash stable partition, and then investigate its algorithmic complexity and suboptimality through various experimental results. To the best of our knowledge, this is the first work that proposes a distributed approach for MinGAP-MR and utilises it for the MRTA problem considered.

5.2 Problem Formulation

Problem 1 (MinGAP-MR)**.** Suppose that there exist a set of n_a agents $\mathcal{A} = \{a_1, ..., a_{n_a}\}$ and a set of n_t tasks $\mathcal{T} = \{t_1, ..., t_{n_t}\}$. For each task t_j, each agent a_i is associated with different *work capacity* w_{ij} and *cost* c_{ij}. Each task t_j has its *minimum workload requirement* R_j to be fulfilled by the aggregated work capacities of multiple agents. Any of the agents is incapable of executing any task alone (i.e., $\max_{\forall a_i \in \mathcal{A}} w_{ij} < R_j$, $\forall t_j \in \mathcal{T}$). Note that R_j, w_{ij} and c_{ij} are non-negative. The objective is to distribute the agents to the tasks in a way that satisfies the demands of the tasks while minimising the total cost of assigned agents:

$$\min_{\{x_{ij}\}} \quad \sum_{i=1}^{n_a} \sum_{j=1}^{n_t} c_{ij} x_{ij} \tag{5.1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n_t} x_{ij} \leq 1, \quad \forall i = 1, 2, ..., n_a \tag{C1}$$

$$\sum_{i=1}^{n_a} w_{ij} x_{ij} \geq R_j, \quad \forall j = 1, 2, ..., n_t \tag{C2}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \tag{C3}$$

where $x_{ij} = 1$ if agent a_i is assigned to task t_j. Constraint (C1) indicates that each agent can be exclusively allocated to at most one task. (C2) ensures that every task's minimum requirement is satisfied. The number of the agents (i.e., n_a) is assumed to be sufficiently large so that there are excessive agents who are not needed for the requirements. Otherwise, there may not exist any feasible solution for the problem.

Table 5.1: Nomenclature

Symbol	Description
$\mathcal{A}$	A set of n_a agents $\{a_1, a_2, ..., a_{n_a}\}$
$\mathcal{T}$	A set of n_t tasks $\{t_1, t_2, ..., t_{n_t}\}$
t_0	the void task (i.e., not to work any task)
w_{ij}	The work capacity of agent a_i with regard to task t_j
c_{ij}	The (original) cost of agent a_i to perform task t_j
c_{ij+}	The learnt cost of agent a_i to perform task t_j (Eqn. (5.3))
λ	The learning rate to affect c_{ij+} (Eqn. (5.3))
R_j	The minimum requirement for task t_j
$\mathcal{S}_j$	The (task-specific) coalition for task t_j
Π	The (disjoint) partition of $\mathcal{A}$, i.e., $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{n_t}, \mathcal{S}_0\}$
$\Pi(i)$	The index of the task assigned to agent a_i, given Π
e_{ij}	The expense of agent a_i for coalition $\mathcal{S}_j$ (Eqn. (5.2))

MinGAP-MR is $\mathcal{NP}$-hard because it is a generalised version of *the 0/1 minimisation knapsack problem* (will be shown in Definition 2), which is also the case [12].

Assumption 1 (*Agents' communication*)**.** The communication network of the entire agents is at least strongly-connected. Given a network, $\mathcal{N}_i$ denotes a set of neighbour agents for agent a_i.

5.3 The Proposed Game-theoretical Approach

5.3.1 Preliminaries

Given $\mathcal{A}$ and $\mathcal{T}$, we define a *partition* as a set $\Pi = \{\mathcal{S}_1, ..., \mathcal{S}_{n_t}, \mathcal{S}_0\}$ such that $\mathcal{S}_j \subseteq \mathcal{A}$, $\forall t_j \in \mathcal{T} \cup \{t_0\}$ and $\mathcal{S}_j \cap \mathcal{S}_k = \emptyset$ for $j \neq k$. Here, t_0 is *the void task* indicating that "not

to work any task in $\mathcal{T}$". $\mathcal{S}_j$ is the (task-specific) *coalition* of agents assigned to task t_j. $\Pi(i)$ indicates the index of the task to which agent a_i is assigned, for example, $\mathcal{S}_{\Pi(i)}$ is the coalition where the agent belongs to. Let e_{ij}, which will be formally defined in Section 5.3.2, denote the *expense* for agent a_i to execute task t_j with other co-workers in $\mathcal{S}_j$.

We model the problem considered as a coalitional game where every agent selfishly seeks to form a coalition that minimises its expense. Note that the expense is not the costs c_{ij} but the virtual currency of the game to coordinate the self-interested agents. The key idea of our proposed approach is that, given a partition Π, an arbitrary agent a_i investigates the expenses for all possible coalitions (i.e., e_{ij}, $\forall \mathcal{S}_j \in \Pi$) and joins the most preferred one. The agent updates the partition information to reflect its new decision, and broadcasts to other agents. Another agent executes the same procedure at the next iteration, and so forth. This iterative process will be terminated if the partition becomes *Nash stable*, where no agent can benefit by its unilateral deviation.

Definition 1 (*Nash stable partition*)**.** A partition Π is said to be *Nash stable* if it holds for every agent $a_i \in \mathcal{A}$ that $e_{i,\Pi(i)} \leq e_{ij}$, $\forall \mathcal{S}_j \in \Pi$.

In general, the existence of a Nash stable partition is not guaranteed. Our previous work [18] proposed a decentralised game-theoretical framework (called *GRAPE (GRoup Agent Partitioning and Placing Event)*) whereby homogeneous agents with *SPAO (Single-Peaked-At-One)* preferences, which can be interpreted as *social inhibition*, are able to find a Nash stable partition. Nevertheless, it can not be applied to the problem considered in this chapter because of the heterogeneity of agents. Convergence of such diverse agents towards a Nash stable partition will be discussed later.

This study will use an algorithm for *the 0/1 minimisation knapsack problem* (MinKP, for short) [12] as its subroutine. MinKP is defined as:

Definition 2 (*MinKP*)**.** Suppose that there are a knapsack with its minimum requirement R and a set of n items $Z = \{z_1, ..., z_n\}$, where each item z_i has its value v_i and cost c_i. The objective is to pack the knapsack with the items so that the total value of all inserted items exceeds the minimum requirement while minimising the resultant total cost:

$$\min_{\{y_i \in \{0,1\}\}} \quad \sum_{i=1}^{n} c_i y_i \qquad \text{s.t.} \quad \sum_{i=1}^{n} v_i y_i \geq R$$

where $y_i = 1$ if item z_i is inserted in the knapsack. Let $\textsc{MinKP}(Z, R, \mathcal{V}, \mathcal{C})$ denote an algorithm for MinKP, where $\mathcal{V} = \{v_1, ..., v_n\}$ and $\mathcal{C} = \{c_1, ..., c_n\}$. The output of this algorithm is the set of selected item for the knapsack.

5.3.2 Design of an Agent's Expense Function

Given the current partition Π, the expense of agent a_i's with regard to each coalition $\mathcal{S}_j \in \Pi \setminus \{\mathcal{S}_0\}$ is defined as follows:

$$e_{ij} := \begin{cases} c_{ij+} & \text{if } a_i \in \hat{\mathcal{S}}_j, \\ \infty & \text{otherwise,} \end{cases} \tag{5.2}$$

where c_{ij+} is the *learnt cost* of agent a_i for task t_j (which will be explain later), and $\hat{\mathcal{S}}_j$ is the set of agents eligible to join the coalition for task t_j. During decision-making process, there must happen the case where the existing coalition $\mathcal{S}_j$ has already superfluous agents to comply the minimum requirement R_j so some of them may be redundant. We set that each agent a_i has to ascertain its *eligibility* for $\mathcal{S}_j$ by an algorithm for MinKP: $\hat{\mathcal{S}}_j := \textsc{MinKP}(\mathcal{S}_j \cup \{a_i\}, R_j, \mathcal{W}_j, \mathcal{C}_j)$[1]; if eligible (i.e., $a_i \in \hat{\mathcal{S}}_j$), then the agent regards expense e_{ij} as learnt cost c_{ij+}.

$\hat{\mathcal{S}}_j$ may differ depending on the existing coalition $\mathcal{S}_j$. For instance, MinKP selects agent a_1 instead of a_2 in some cases, vice versa in the other cases. It was observed that this fact sometimes prevent the agents from converging to a Nash stable partition.

Such possible divergence can be addressed by utilising *learnt costs* instead of original costs. The learnt cost of agent a_i with regard to task t_j is updated, while the agent is learning, as:

$$c_{ij+} \leftarrow c_{ij+} + \lambda \cdot c_{ij} \tag{5.3}$$

where $\lambda > 0$ is the *learning rate*. Initially, $c_{ij+} = c_{ij}$. Whenever agent a_i changes its decision from task t_j to another (or possibly the void task), the agent *learns* that the previously chosen task is not suitable for itself and penalises the task gradually by the learning rate as Equation (5.3). This can be called as *tabu learning* [19]. This

[1]Please refer to Definition 2. Here, $\mathcal{W}_j = \{w_{kj} \mid \forall a_k \in \mathcal{S}_j \cup \{a_i\}\}$ and $\mathcal{C}_j = \{c_{kj} \mid \forall a_k \in \mathcal{S}_j \setminus \{a_i\}\} \cup \{c_{ij+}\}$.

iterative decision-making and learning process develops the partition of agents as well as their learnt costs, by which conflicts between the agents can be resolved and a Nash stable partition can be determined (the proof is provided in Section 5.4.1). When the process terminates, the resultant objective function value is calculated as Equation (5.1) by using original costs c_{ij} instead of expenses in Equation (5.2), which are auxiliary variables only to find a conflict-free assignment.

Note that if every expense for task $t_j \in \mathcal{T}$ is infinity, then the agent chooses the void task.

5.3.3 Decentralised Algorithm

Our proposed decision-making procedure is presented as Algorithm 1. Given the current locally-known partition[2], denoted by Π^i, agent a_i investigates every coalition $\mathcal{S}_j \in \Pi^i$ and computes the corresponding expense e_{ij} according to Equation (5.2) (Lines 5–10). If the least expense value provides infinity, then the agent set the preferred coalition index j^* as zero (Lines 12–13), meaning that it will move to $\mathcal{S}_0$. If the preferred coalition is not the existing one, then the agent updates the learnt cost for the existing coalition, joins to $\mathcal{S}_{j^*}$, amends Π^i to reflect its new decision, increases r^i (which is the number of evolutions of the partition), and generates a new random time stamp s^i (Lines 15–21). Then, the agent constructs a message $\mathsf{msg}^i := \{r^i, s^i, \Pi^i, \mathsf{satisfied}^i\}$ and sends it to other neighbour agents, and vice versa (Line 24). Amongst these multiple partition information, only one can be distributedly chosen by any agent at last through the distributed mutex algorithm [18] in Appendix, denoted by D-Mutex (Line 26). The distributed mutex algorithm enables Algorithm 1 to be executed asynchronously and distributedly as long as the network of the given agents is at least strongly-connected. Note that the agent shares only msg^i with other agents, keeping the learnt costs locally.

When updating Π^i (Line 17), agent a_i might be allowed to amend the statuses of other agents who will be expelled by the agent (i.e., $\forall a_k \in \mathcal{S}_{j^*} \setminus \hat{\mathcal{S}}_{j^*}$). However, this setting certainly results in an increase of the corresponding communication transactions at every iteration. To avoid this, Algorithm 1 is designed such that agent a_i only notifies

[2]As the algorithm is decentralised, the current partition information may be differently known by different agents. Thus, we let Π^i denote the locally-known information of agent a_i.

its decision change to others. Although this might temporarily mislead some agents to assume that they are still eligible for the existing coalitions, they can eventually notice their ineligibilities (please refer to Section 5.4.1) if the MinKP algorithm in Line 8 holds the following condition:

Condition 1. Suppose that $\hat{\mathcal{S}} = \textsc{MinKP}(\mathcal{S}, ...)$. The algorithm holds that $\hat{\mathcal{S}} = \textsc{MinKP}(\hat{\mathcal{S}} \cup \mathcal{S}^{\circ}, ...)$, $\forall \mathcal{S}^{\circ} \subseteq \mathcal{S} \setminus \hat{\mathcal{S}}$.

This chapter utilises the approximation algorithm "MinGreedy" in [12], which holds this condition.

5.4 Analysis

5.4.1 Convergence to a Nash stable Partition

For simplicity of description, this section assumes that the agents are communicationally fully-connected and accessible to the shared memory, where the current partition Π is stored. At each iteration, a single agent exclusively updates the partition information if necessary.

We first show that it is enough for agent a_i in Algorithm 1 to notify only its decision change to others. Suppose that agent a_i decides to join the coalition for task t_j and thereby a_l must be expelled from the coalition (i.e., $a_l \in \mathcal{S}_j \setminus \hat{\mathcal{S}}_j$). At the next iteration, in the case that agent a_l executes MinKP holding Condition 1, the agent realises that itself is not supposed to be in the existing coalition, and then leaves from there.

In the other case, another arbitrary agent a_m misconceiving that agent a_l is still in $\mathcal{S}_j$ executes Algorithm 1, and may also join the coalition for task t_j. When agent a_m obtains $\hat{\mathcal{S}}_j = \textsc{MinKP}(\mathcal{S}_j \cup \{a_i\} \cup \{a_m\}, ...)$, $\hat{\mathcal{S}}_j$ may include a_l because the combination of a_l and a_m may be preferred than some existing agent (i.e., the agents' eligibilities for the coalition may be changed depending on the input agent set). As an extreme case, suppose that all the other agents consequently join the coalition without expelling any of all the existing agents in this way, and the lastly joined agent obtains $\hat{S}_j^{\star} = \textsc{MinKP}(\mathcal{A}, ...)$. From the next iteration, agents who are supposed to be expelled (i.e.,

Algorithm 1 Decision-making of agent a_i

// Initialisation

$\mathsf{satisfied}^i \leftarrow \mathbf{0}_{n_a \times 1}$; $r^i \leftarrow 0$; $s^i \leftarrow 0$

$\Pi^i :=$ the initial partition

// Decision-making process begins

while $\mathsf{satisfied}^i \neq \mathbf{1}_{n_a \times 1}$ **do**

Make a new decision if necessary

  if $\mathsf{satisfied}^i[i] = 0$ **then**

    for each coalition $\mathcal{S}_j \in \Pi^i \setminus \{\mathcal{S}_0\}$ **do**

      $\mathcal{W}_j = \{w_{kj} \mid \forall a_k \in \mathcal{S}_j \cup \{a_i\}\}$

      $\mathcal{C}_j = \{c_{kj} \mid \forall a_k \in \mathcal{S}_j \setminus \{a_i\}\} \cup \{c_{ij+}\}$

      $\hat{\mathcal{S}}_j = \text{MINKP}(\mathcal{S}_j \cup \{a_i\}, R_j, \mathcal{W}_j, \mathcal{C}_j)$

      Compute e_{ij} using Eqn. (5.2)

    end for

    $j^* = \text{argmin}_{\forall j} e_{ij}$; $e^* = \min_{\forall j} e_{ij}$;

    if $e^* = \infty$ **then** *// No coalition is preferred*

      $j^* := 0$

    end if

    if $j^* \neq \Pi^i(i)$ **then**

      $c_{i,\Pi^i(i)+} = c_{i,\Pi^i(i)+} + \lambda \cdot c_{i,\Pi^i(i)}$

      Join $\mathcal{S}_{j^*}$ and update Π^i

      $r^i \leftarrow r^i + 1$

      $s^i \in \text{unif}[0, 1]$

      $\mathsf{satisfied}^i \leftarrow \mathbf{0}_{n_a \times 1}$

    end if

    $\mathsf{satisfied}^i[i] = 1$

  end if

// Broadcast the local information to neighbour agents

  Broadcast $\mathsf{msg}^i = \{r^i, s^i, \Pi^i, \mathsf{satisfied}^i\}$ and receive msg^k from its neighbours $\forall a_k \in \mathcal{N}_i$

// Select the valid partition from all the received messages

  Collect all the messages $\mathcal{M}^i_{rcv} = \{\mathsf{msg}^i, \forall \mathsf{msg}^k\}$

  $\{r^i, s^i, \Pi^i, \mathsf{satisfied}^i\} :=$ D-MUTEX$(\mathcal{M}^i_{rcv})$

end while

$\forall a \in \mathcal{A} \setminus \hat{S}_j^\star$) eventually notice their ineligibilities for task t_j because MinKP holds Condition 1.

Theorem 1. *Given that the communication of the agents is at least strongly-connected, Algorithm 1 terminates and converges to a Nash stable partition within a finite number of iterations.*

Proof. Assuming the shared memory, we will first show the convergence of the centralised version algorithm, and then relax it for the decentralised version.

Suppose that the statement of the theorem is false, and there is an agent a_∞ who infinitely changes its decision amongst a set of task $\mathcal{T}_\infty \subseteq \mathcal{T}$, preventing convergence to a Nash stable partition. Whenever the agent revises its decision from a task, the learnt cost for the task is increased by the tabu learning (Line 16). In the case where $\mathcal{T}_\infty \subset \mathcal{T}$, all the learnt costs for tasks in $\mathcal{T}_\infty$ finally become more expensive than those for any tasks in $\mathcal{T} \setminus \mathcal{T}_\infty$ as the agent travels within $\mathcal{T}_\infty$. Thus, the agent will eventually join $\mathcal{S}_j$ for any task $t_j \in \mathcal{T} \setminus \mathcal{T}_\infty$, which contradicts our supposition.

If $\mathcal{T}_\infty = \mathcal{T}$, agent a_∞ finally cannot select any task in $\mathcal{T}$ and then join $\mathcal{S}_0$ (Lines 11–13), because the agent rules out itself due to the increased learnt costs and the expense function in Equation (5.2). This fact also contradicts our supposition, which proves the convergence of the centralised version algorithm.

Let us now turn to the setting where there is not shared memory and the agents are strongly-connected. As is shown in Algorithm 1, consider that at each iteration, every agent locally updates its locally-known partition and broadcasts the updated information to its neighbour agents, simultaneously. From the view point of each partition, it can be seen that the partition information is somehow circulated amongst and updated by a sequence of the agents as if a centralised mutex algorithm supports this process based on the fully-connected network and the shared memory. Hence, as we proved before, each partition finally evolves to be Nash stable, but we have multiple Nash stable partitions. Thanks to the subroutine D-Mutex, however, only one Nash stable partition is eventually chosen (such a partition is said to be *valid*) by the entire agents in a decentralised and asynchronous manner as long as their communication is at least strongly-connected [18]. □

Remark 1. The proof of Theorem 1 implies that the idea of allowing agents to be selfish

and addressing their possible conflicts by tabu-learning heuristics always guarantees convergence towards a Nash stable partition, regardless of agents' different interests and whether they are homogeneous or heterogeneous. For the rest of this thesis, we refer to this approach as *T-GRAPE (Tabu-learning-based GRAPE)*.

Since it is assumed that there are sufficient agents for a mission, there must be some agents in $\mathcal{S}_0$. This implies that a Nash stable partition provides a feasible assignment satisfying all the constraints (C1)–(C3).

5.4.2 Computation & Communication Complexities

The computational complexity of Algorithm 1 for each agent per iteration can be said to be $O(\sum_{\forall t_j \in \mathcal{T}} f(|\mathcal{S}_j|+1))$, where $f(n)$ is the running time for MinKP when n items (i.e., agents) are given as inputs. This complexity is lower than just $O(f(|\mathcal{A}|))$ if $f(n)$ is super-additive (i.e., $f(n_1+n_2) \geq f(n_1)+f(n_2)$). This is the case for MinKP used in this chapter, i.e., $O(n\ln(n))$ [12]. Additionally, in practice, there may be some redundant agents who belong to $\mathcal{S}_0$. This fact also makes the complexity of the proposed algorithm much less than $O(f(|\mathcal{A}|))$. Overall, it can be conservatively said that the convergence time of the proposed approach is $O(|\mathcal{T}| \cdot f(|\mathcal{S}_{max}|) \cdot C)$. Here, $\mathcal{S}_{max} = \text{argmax}_{\forall \tau, \forall \mathcal{S}_j \in \Pi \setminus \{\mathcal{S}_0\}} |\mathcal{S}_j|$ is the maximum-size coalition until convergence except $\mathcal{S}_0$, where τ indicates each iteration happened. C is the number of iterations until convergence.

The required communicational traffic for an agent at each iteration is $O(|\mathcal{A}|)$, as the agent needs to notify only its status change to at most $|\mathcal{A}|-1$ of other agents even in a multi-hop fashion. Thus, the overall communicational complexity is $O(|\mathcal{A}| \cdot C)$.

The number of required iterations C may vary depending on the communication network. This effect will be experimentally investigated in the following section.

5.5 Empirical Validation

This section performs numerical experiments to validate the characteristics of the proposed approach regarding its suboptimality and the number of required iterations until

converging to a Nash stable partition on a strongly-connected (SC) network or the fully-connected (FC) network of agents.

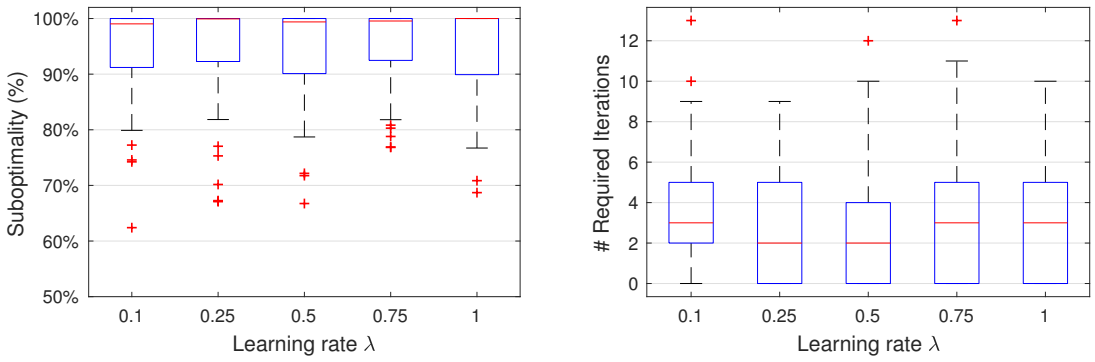
For small-sized instances ($n_a = 10$, $n_t = 3$), we conduct 100 runs of Monte-Carlo experiments and investigate the proposed algorithm, compared with the optimal results obtained by a brute-force algorithm. For each instance, the parameters for agent a_i and task t_j are uniform-randomly generated from the following ranges: $R_j \in [5, 10]$; $c_{ij} \in [0.1, 1]$; and $w_{ij} \in [n_t/n_a, 0.9] \times R_j$. Moreover, in order to identify the effect of the learning rate, it is set as $\lambda \in \{0.1, 0.25, 0.5, 0.75, 1\}$. For each instance, a strongly-connected network is randomly generated.

For the rest of this experiment, we set an initial assignment as follows. Every agent firstly chooses the task that gives the least expense without consideration of other agents. Based on this partition, the agent checks its eligibility to stay the most preferred coalition, and goes to $\mathcal{S}_0$ if it is not eligible. The resultant partition from this process is the initial assignment, and then an arbitrary agent begins to execute Algorithm 1.

Figure 5.1(a) shows that the mean value of the suboptimality is approximately 95% and its standard deviation is less than 8%, regardless of the learning rate. In Figure 5.1(b), it is also presented that the number of required iterations until convergence is irrespective of the learning rate.

We also conduct Monte-Carlo experiments for large-sized instances where $(n_t, n_a) \in \{(5, 50), (5, 100)\}$. The problem-related parameters are drawn as follows: $c_{ij} \in [c_{\min}, 1]$; $R_j \in [5, 10]$; and $w_{ij} \in [n_t/n_a, w_{\max}] \times R_j$. We also variously set $c_{\min} \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and $w_{\max} \in \{n_t/n_a, 0.2, 0.3, 0.5, 0.7, 0.9\}$. For every combinational case of $c_{\min}$ and $w_{\max}$, 100 instances are uniform-randomly generated from the aforementioned ranges. Here, we set λ to 0.25. We also perform the same experiments under the fully-connected communication network.

For such large-sized instances, since it is not feasible to obtain the optimal solution within a reasonable computation time, we instead utilise the *Linear-and-Conflict Relaxation* (LCR) solution to analyse the suboptimality of the proposed approach. The LCR solution is the outcome when (C1) and (C3) are relaxed. This can be simply determined as follows: fill every knapsack (i.e., task) with all the given items (i.e., agents) in a greedy manner until (C2) is satisfied no matter which items are already assigned

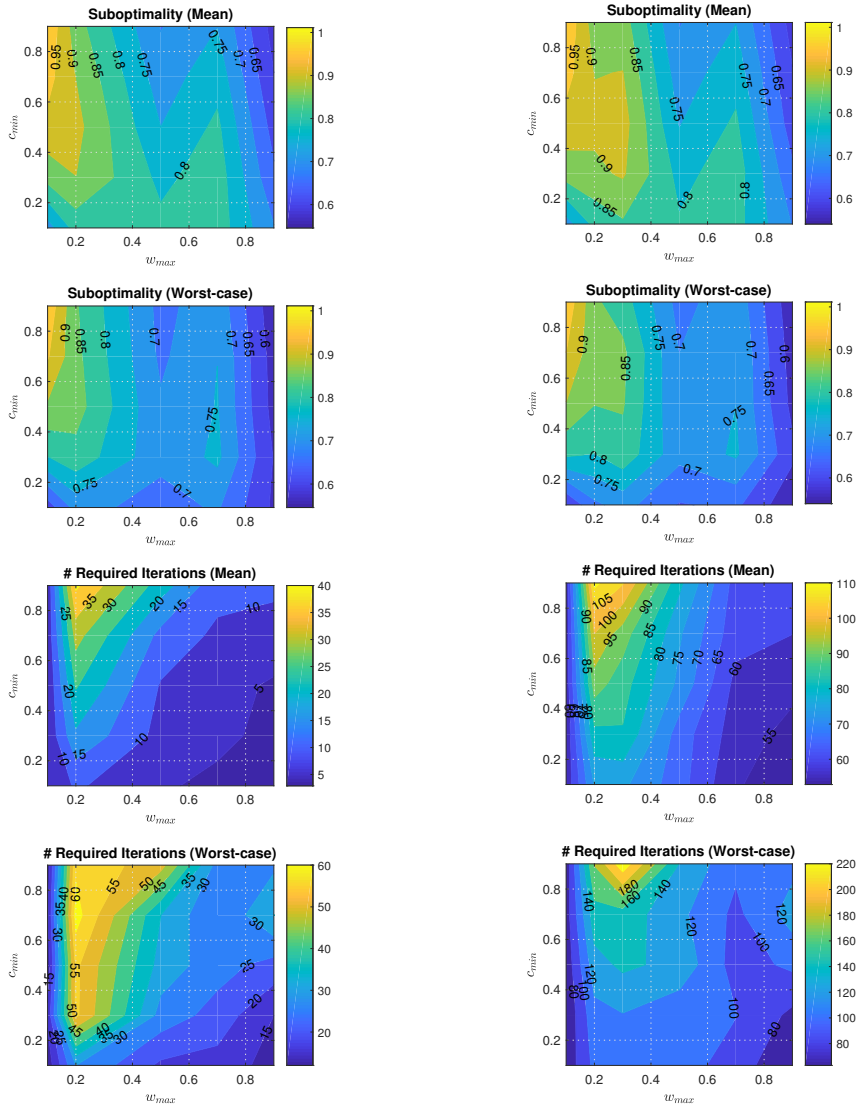


(a) (True) Suboptimality **(b)** # Required Iterations

Figure 5.1: Parametric analysis regarding the learning rate λ ($n_a = 10$, $n_t = 3$) under a strongly-connected network: (a) the suboptimality (i.e., J_{OPT}/J_A, where J_{OPT} is the total cost by a brute-force algorithm and J_A is that by the proposed algorithm); (b) the number of iterations required for convergence to a Nash stable partition.

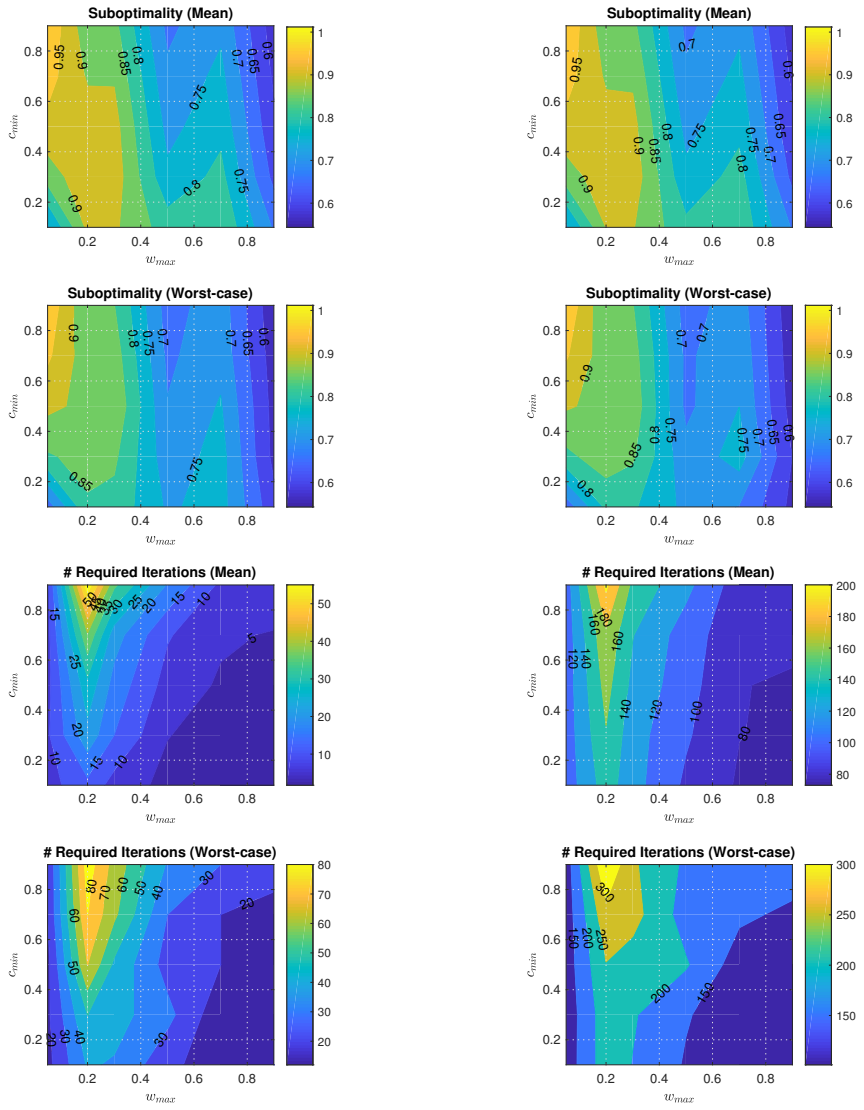
to other knapsacks (i.e., conflict-relaxed); and the last inserted item can be included fractionally (i.e., linear-relaxed). It is obvious that $J_{LCR} \leq J_{OPT}$, where J_{LCR} denotes the objective function value of the LCR solution. From this, the suboptimality of the proposed algorithm can be conservatively investigated by comparing with J_{LCR} instead of J_{OPT}, because the true suboptimality (i.e., J_{OPT}/J_A) is lower-bounded by J_{LCR}/J_A, which we call *the quasi-suboptimality*. This approach is inspired from [20], where greedy algorithms for the $0/1$ single knapsack problem are investigated by comparing with the linear relaxation solution and the optimal solution.

Figures 5.2 and 5.3 show the mean and the worst values regarding the quasi-suboptimality and the number of required iterations, amongst 100 instances at each combinational case of $w_{\max}$ and $c_{\min}$ under either the fully-connected network or a strongly-connected network. Overall, the quasi-suboptimality is at least more than 50% at any of the cases, as shown in the-second-row subfigures. It is presented that as $w_{\max}$ is decreasing and $c_{\min}$ is increasing, the quasi-suboptimality becomes higher. One possible explanation is that agents in this setting become almost homogeneous with having much smaller w_{ij} compared with R_j. This tendency makes the problem trivial so that a near-optimal solution can be found by just assigning arbitrary agents to any tasks. The-third-row and the-fourth-row subfigures illustrate that the number



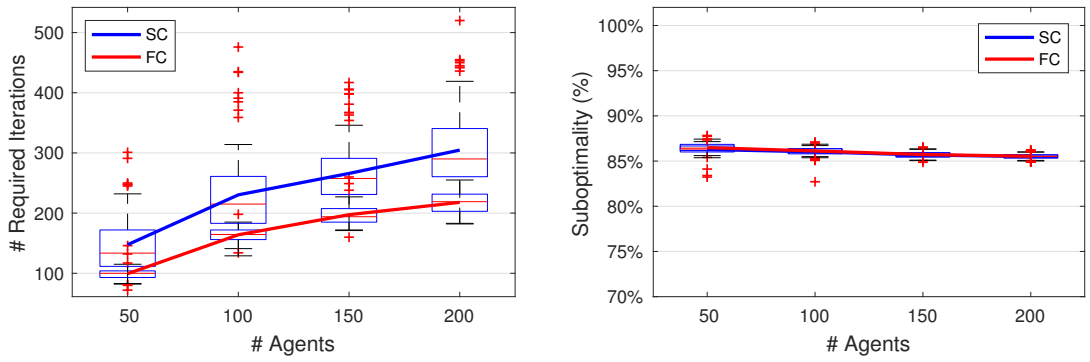
(a) $n_t = 5, n_a = 50$ (FC) **(b)** $n_t = 5, n_a = 50$ (SC)

Figure 5.2: Contour plots of the quasi-suboptimality and the number of iterations required for convergence to a Nash stable partition at every combinational case of $w_{\max}$ and $c_{\min}$ under either the fully-connected network (FC) or a strongly-connected network (SC). The-first-row and the-second-row subfigures show the mean and the minimum values of the quasi-suboptimality, and the-third-row and the-fourth-row subfigures show the average and the maximum values of the number of required iterations, respectively, amongst 100 uniform-randomly-generated instances at each case.



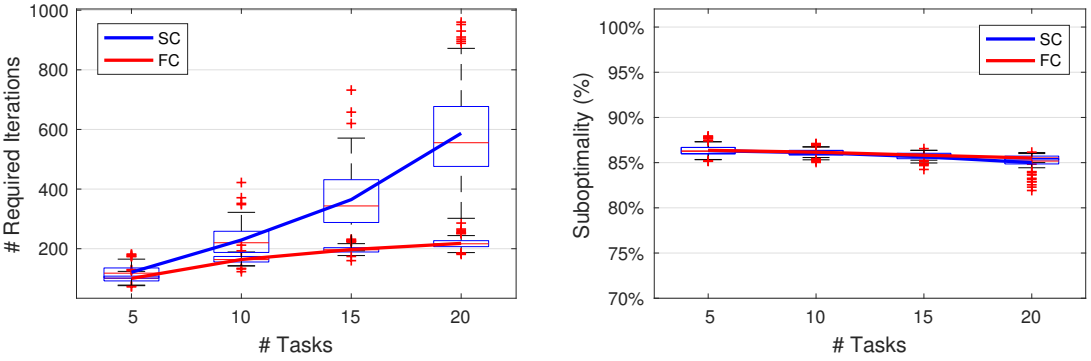
(a) $n_t = 5, n_a = 100$ (FC) **(b)** $n_t = 5, n_a = 100$ (SC)

Figure 5.3: Contour plots of the quasi-suboptimality and the number of iterations required for convergence to a Nash stable partition at every combinational case of $w_{\max}$ and $c_{\min}$ under either the fully-connected network (FC) or a strongly-connected network (SC). The-first-row and the-second-row subfigures show the mean and the minimum values of the quasi-suboptimality, and the-third-row and the-fourth-row subfigures show the average and the maximum values of the number of required iterations, respectively, amongst 100 uniform-randomly-generated instances at each case.



(a) # Required Iterations **(b)** Quasi-suboptimality

Figure 5.4: Parametric analysis regarding various n_a (with fixed $n_t = 10$)



(a) # Required Iterations **(b)** Quasi-suboptimality

Figure 5.5: Parametric analysis regarding various n_t (with fixed $n_a = 100$)

of required iterations stays at a similar level of n_a in the FC test cases, but increases in the SC test cases.

We further investigate the algorithmic complexity and the quasi-suboptimality with various n_a and n_t. The results of 100 simulation runs at $c_{\min} = 0.9$ and $w_{\max} = 0.4$ are provided in Figure 5.4 and 5.5. In all the test cases, more iterations are required under SC than FC. This is because agents being less-connected may induce unnecessary iterations that would not have happened if they were fully-connected [18]. The number of required iterations moderately increases with respect to n_a or n_t under FC, whereas, under SC, the algorithmic complexity is shown to be more affected by n_t. A possible explanation is such that as n_t/n_a increases, each agent has more options to choose and

may have more tabu-learning process, especially under a strongly-connected network.

On the other hand, the quasi-suboptimality (i.e., J_{LCR}/J_A) is slightly reduced by increasing n_a or n_t, but still stays reasonable. This little decrease may be caused by J_{LCR}, which can be decreased as more agents or tasks are given. Another finding is that the quasi-suboptimality is hardly affected by communication network, as is the case in [18].

5.6 Conclusion

This chapter studied the MRTA problem where each task has its minimum workload requirement to be fulfilled by multiple heterogeneous agents. The objective is to find an assignment that minimises the total cost of assigned agents while satisfying the tasks' requirements. We proposed a distributed game-theoretical approach, where each agent selfishly joins the most favourite coalition, and showed that a Nash stable partition can always be determined with tabu-learning heuristics. Our experimental results revealed that the suboptimality of the proposed algorithm is at least more than 50%, and the number of required iterations until convergence remains the same order of the number of given agents at various problem settings.

More research is required to analytically evaluate the suboptimality of the proposed algorithm, especially connecting with the approximation ratio of its subroutine for MinKP. Furthermore, a natural progression of this work is to study how to exploit finally-unassigned agents depending on the context of a given mission. It would also be interesting to address a mixed mission scenario where there are additionally some robots capable of executing a task alone.

Appendix

For Algorithm 1, we use the distributed mutex algorithm (i.e., Algorithm 2) proposed in our previous work [18] as a subroutine. The algorithm makes sure that there is only one (local) partition that dominates (or will finally dominate depending on the communication network) any other partitions. In other words, multiple partitions locally evolve and some of them only eventually can survive even under asynchronous behaviours of

Algorithm 2 Distributed Mutex Subroutine [18]

function D-MUTEX($\mathcal{M}^i_{rcv}$)
  for each message $\mathsf{msg}^k \in \mathcal{M}^i_{rcv}$ **do**
    if $(r^i < r^k)$ or $(r^i = r^k \ \& \ s^i < s^k)$ **then**
      $r^i \leftarrow r^k$
      $s^i \leftarrow s^k$
      $\Pi^i \leftarrow \Pi^k$
      $\mathsf{satisfied}^i \leftarrow \mathsf{satisfied}^k$
    end if
  end for
  return $\{r^i, s^i, \Pi^i, \mathsf{satisfied}^i\}$
end function

agents as long as their communication network is at least strongly-connected. Even if we may encounter multiple Nash stable partitions at last, one of them can be distributedly selected by the agents.

References

[1] H.-S. Shin and P. Segui-Gasco, "UAV Swarms: Decision-Making Paradigms," in *Encyclopedia of Aerospace Engineering*. Chichester, UK: John Wiley & Sons, Ltd, dec 2014, pp. 1–13.

[2] K. Barton and D. Kingston, "Systematic Surveillance for UAVs: A Feedforward Iterative Learning Control Approach," in *American Control Conference*, Washington, DC, USA, 2013, pp. 5917–5922.

[3] I. Bekmezci, O. K. Sahingoz, and S. Temel, "Flying Ad-Hoc Networks (FANETs): A Survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[4] I. Jang, J. Jeong, H.-S. Shin, S. Kim, A. Tsourdos, and J. Suk, "Cooperative Control for a Flight Array of UAVs and an Application in Radar Jamming," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8011–8018, 2017.

[5] B. P. Gerkey and M. J. Matarić, "A Formal Analysis and Taxonomy of Task Allocation in Multi-robot Systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[6] T. Öncan, "A Survey of the Generalized Assignment Problem and Its Applications," *INFOR: Information Systems and Operational Research*, vol. 45, no. 3, pp. 123–141, 2007.

[7] T. William Mather and M. Ani Hsieh, "Distributed Robot Ensemble Control for Deployment to Multiple Sites," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, 2011.

[8] P. Segui-Gasco, H.-S. Shin, A. Tsourdos, and V. J. Segui, "Decentralised Submodular Multi-Robot Task Allocation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hamburg, Germany, 2015, pp. 2829–2834.

[9] H. L. Choi, L. Brunet, and J. P. How, "Consensus-based Decentralized Auctions for Robust Task Allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[10] L. Luo, N. Chakraborty, and K. Sycara, "Distributed Algorithms for Multirobot Task Assignment With Task Deadline Constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 876–888, 2015.

[11] L. Luo, N. Chakraborty, and K. Sycara, "Provably-Good Distributed Algorithm for Constrained Multi-Robot Task Assignment for Grouped Tasks," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 19–30, 2015.

[12] M. M. Güntzer and D. Jungnickel, "Approximate Minimization Algorithms for the 0/1 Knapsack and Subset-Sum Problem," *Operations Research Letters*, vol. 26, no. 2, pp. 55–66, 2000.

[13] X. Han and K. Makino, "Online Minimization Knapsack Problem," in *Approximation and Online Algorithms. WAOA 2009. Lecture Notes in Computer Science.* Springer, Berlin, Heidelberg, 2010, vol. 5893, pp. 182–193.

[14] M. J. Geiger and W. Wenger, "On the Assignment of Students to Topics: A Variable Neighborhood Search Approach," *Socio-Economic Planning Sciences*, vol. 44, no. 1, pp. 25–34, 2010.

[15] M. Bender, C. Thielen, and S. Westphal, "A Constant Factor Approximation for the Generalized Assignment Problem with Minimum Quantities and Unit Size Items," in *Mathematical Foundations of Computer Science 2013. MFCS 2013. Lecture Notes in Computer Science, vol 8087.* Berlin: Springer, 2013, pp. 135–145.

[16] S. O. Krumke and C. Thielen, "The Generalized Assignment Problem with Minimum Quantities," *European Journal of Operational Research*, vol. 228, no. 1, pp. 46–55, 2013.

[17] Z. Han, D. Niyato, W. Saad, T. Basar, and A. Hjørungnes, *Game Theory in Wireless and Communication Networks : Theory, Models, and Applications.* Cambridge University Press, 2012.

[18] I. Jang, H.-S. Shin, and A. Tsourdos, "Anonymous Hedonic Game for Task Allocation in a Large-scale Multiple Agent System," *IEEE Transactions on Robotics (in press)*, 2018.

[19] D. Beyer and R. Ogier, "Tabu Learning: A Neural Network Search Method for Solving Nonconvex Optimization Problems," in *IEEE International Joint Conference on Neural Networks*, 1991, pp. 953–961.

[20] N. N. Galim'yanova, A. A. Korbut, and I. K. Sigal, "Ratios of Optimal Values of Objective Functions of the Knapsack Problem and Its Linear Relaxation," *Journal of Computer and Systems Sciences International*, vol. 48, no. 6, pp. 906–913, 2009.

Chapter 6

An Integrated Decision-making Framework of a Robotic Swarm

6.1 Introduction

A networked system of a large number of aerial robots, called *(aerial) robotic swarm* or *UAV swarm*, has been attracting many researchers' interest because of its promising advantages such as its versatility, robustness, and adaptiveness [1–3]. To name a few, possible applications of UAV swarms include environmental monitoring [4], ad-hoc network relay [5], disaster management [6], and cooperative military missions [7]. However, there still exist various technical challenges to realise a robotic swarm into real-life applications: for example, in view of autonomous decision-making domain, multi-robot task allocation problem [8–10], and trajectory optimisation including collision avoidance [11, 12].

Cooperation is essential for a robotic swarm: robots (or *agents*) with cheaper components can be realised through mass production in lower cost, but each of them eventually has limited capability to complete a single task alone [13]. Given multiple tasks spatially distributed, the robots have to partition themselves into disjoint task-specific teams (or *coalitions*). This decision-making issue is referred to as *coalition formation* problem [14] or *ST-MR multi-robot task allocation* problem [8]. Even though the robotic swarm was identically manufactured, the agents may be heterogeneous in terms of the

currently-available work resources because, for example, it is probably not possible to charge every robot's battery fully before a mission due to the large cardinality. Such heterogeneity makes the coalition formation problem more complicated. Moreover, the tasks may have their minimum workload requirements, which have to be cooperatively fulfilled by the agents' resources. Having extra resources, in some cases, it is desirable to equitably distribute them in proportion to the minimum requirements to improve the system's fault tolerance.

When implementing a robotic swarm into a mission, we encounter further decision-making issues on top of the task allocation problem. Each task-specific team of robots also have to make decisions regarding who to go which (spatial) work position specifically, called *position allocation* problem (note that a coalition formation outcome only can assign a certain sector for working). Besides, the agents' trajectories towards the assigned positions should be optimised, guaranteeing that any collision with other agents or obstacles will not happen during the mission. What makes the entire problem more involved is that the trajectories affect the expected working resources of the agents when they arrive at and execute their assigned tasks. Therefore, such effects should also be considered in the coalition formation and position allocation problems, and thereby the combined decision-making problem becomes much more complicated than either of individual problems.

Over the last few years, some researchers have begun to address problems of simultaneously finding robot-task assignments and trajectories. One typical strategy is to regard such a complex decision-making issue as a single optimisation problem (e.g., minimising the sum of distance travelled by all the agents towards their assigned positions) and solve it either in a centralised [15] or decentralised manner [16]. In particular, Turpin et al. [16] show that, supposing homogeneous robots, the optimal assignment minimising velocity-squared trajectories without considering collision avoidance can be actually collision-free, and that a suboptimal outcome can be obtained in a decentralised manner. However, one drawback of this work is that a robot's trajectory is assumed as a straight line, which may fail to deal with obstacle avoidance. Alternatively, there have also been decoupling strategies, where tasks are first assigned and then trajectories are planned [17–19]. This type of approaches is advantageous in the sense that each subproblem can be substituted as required depending on practical applications

considered. The existing works [18, 19] utilise for task allocation a stochastic-policy-based method and an auction-based method, respectively, and exploit an MPC-SCP (Model Predictive Control and Sequential Convex Programming) algorithm for path planning [20], which was experimentally shown to be implementable for a multi-robot system on a real-time basis as well as has the potential to consider obstacle avoidance. However, all of the existing studies reviewed consider neither heterogeneous robots nor fair allocation concerning task requirements, which will be addressed in this chapter.

We first formulate all the prescribed decision-making issues of a heterogeneous robotic swarm as a combined optimisation problem, and then propose an integrated framework that addresses the problem in a decentralised fashion. Our approach approximates and decouples the problem into three subproblems, i.e., coalition formation, position allocation, and path planning, which are sequentially addressed by three different subroutine algorithms. Due to the large cardinality of a robotic swarm system, directly obtaining robot-to-position assignment can be restrictive regarding computational complexity. To avoid this hinderance, the proposed approach firstly partitions the robots into disjoint task-specific coalitions, followed by dealing with the position allocation subproblem for the robots assigned to the same task. For the coalition formation subproblem, we propose a game-theoretical method in which each agent unilaterally selects a task-specific coalition with consideration of fair allocation regarding the given tasks' requirements. Then, we show that, given reasonable assumptions, the position allocation subproblem can be efficiently addressed in terms of computational complexity even using a simple sorting algorithm. For the trajectory optimisation, we utilise an MPC-SCP algorithm because of its aforementioned advantages (e.g., real-time implementability). As a proof of concept, we implement the framework into a cooperative stand-in jamming mission scenario using a swarm of micro UAVs and show its feasibility, fault tolerance, and near-optimality based on numerical experiment.

This chapter is organised as follows. Section 6.2 defines the original complex problem, which is decoupled into the three subproblems in Section 6.3. Then, Sections 6.4, 6.5, and 6.6 propose the subroutines for the coalition formation, the position allocation, and the path planning subproblems, respectively. In Section 6.7, we propose the integrated framework consisting of the three subroutines and discuss its properties. Section 6.8 shows the results of numerical experiments using the framework on a UAV swarm's

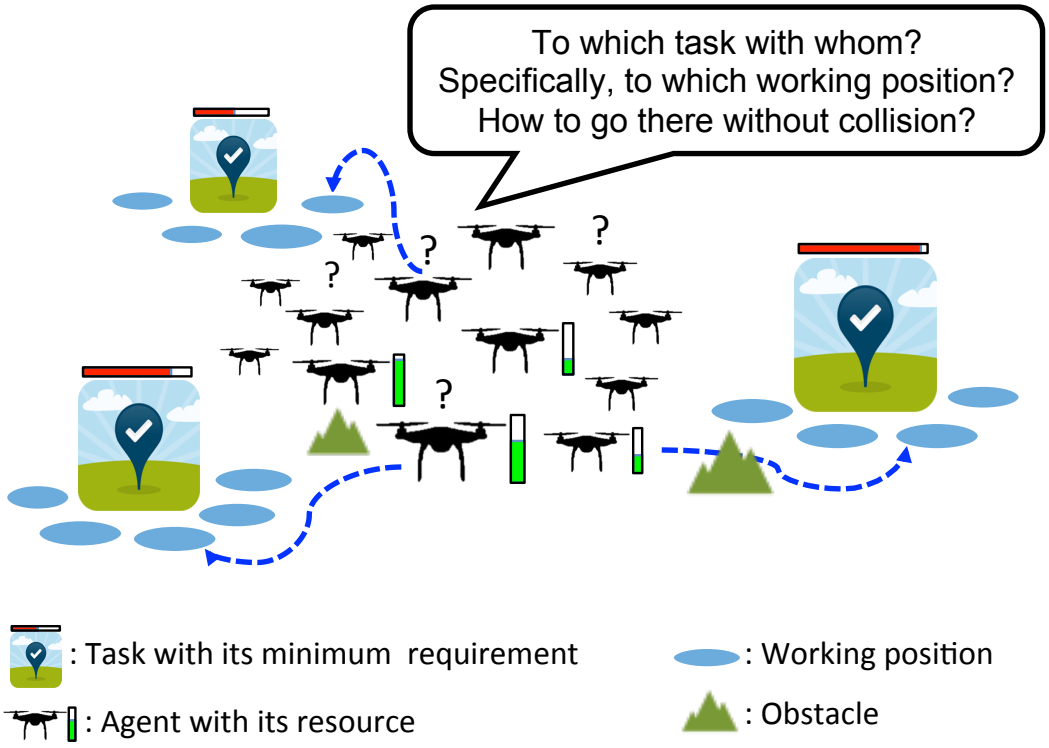


Figure 6.1: A brief illustration of the decision-making issues considered

cooperative stand-in jamming mission.

6.2 Problem Statement: The Original Problem

We have a set of *agents* $\mathcal{A} = \{a_1, a_2, ..., a_{n_a}\}$, and a set of a fewer number of spatially distributed *tasks* $\mathcal{T} = \{t_1, t_2, ..., t_{n_t}\}$, where $n_t < n_a$, to be collaboratively executed by the agents. Each task t_j has a set of (spatial) *work positions* $\mathcal{P}_j = \{p_{m_1}, p_{m_2}, ...\}$, where every $p_m \in \mathbb{R}^{n_d \times 1}$ is an n_d-dimensional Euclidean-space position vector at which an agent can execute the task. Each work position is only associated with a single task, i.e., $\mathcal{P}_{j_1} \cap \mathcal{P}_{j_2} = \emptyset$ for $\forall j_1 \neq j_2$. Without loss of generality, let $\mathcal{P}_1 = \{p_1, ..., p_{|\mathcal{P}_1|}\}$, $\mathcal{P}_2 = \{p_{|\mathcal{P}_1|+1}, ..., p_{|\mathcal{P}_1|+|\mathcal{P}_2|}\}$,..., $\mathcal{P}_{n_t} = \{p_{|\mathcal{P}_{n_t-1}|+1}, ..., p_{|\mathcal{P}_{n_t-1}|+|\mathcal{P}_{n_t}|}\}$. We denote the set of the entire work positions by $\mathcal{P} := \cup_{\forall t_j \in \mathcal{T}} \mathcal{P}_j$. Furthermore, we have a set of (static) *obstacles* $\mathcal{O} = \{o_1, o_2, ..., o_{n_o}\}$ the agents have to avoid during the mission, where each

Table 6.1: Nomenclature

Symbol	Description
$\mathcal{T}$	A set of n_t tasks $\{t_1, t_2, ..., t_{n_t}\}$
$\mathcal{A}$	A set of n_a agents $\{a_1, a_2, ..., a_{n_a}\}$
$\mathcal{S}_j$	The task-specific coalition of agents for task t_j
Π	The set of coalitions $\Pi = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{n_t}, \mathcal{S}_0\}$
$\mathcal{O}$	A set of n_o obstacles $\{o_1, o_2, ..., o_{n_o}\}$
$\mathcal{P}_j$	A set of working positions $\{p_{m_1}, p_{m_2}, ...\}$ where agents can execute task t_j
$\mathcal{P}$	The set of the entire working positions i.e., $\mathcal{P} := \cup_{\forall t_j \in \mathcal{T}} \mathcal{P}_j$
w_{im}	Agent a_i's (expected) work resource at position p_m
c_{im}	The cost if agent a_i works at position p_m
$\bar{w}_{ij}$	Agent a_i's abstracted (expected) work resource for task t_j (regardless of working positions in $\mathcal{P}_j$)
$\bar{c}_{ij}$	The abstracted cost of agent a_i for task t_j (regardless of working positions in $\mathcal{P}_j$)
$w_{i,0}$	Agent a_i's initial work resource
$p_{j,0}$	task t_j's central position
R_j	The minimum requirement for task t_j to be completed
α_j	RSI (Requirement Satisfaction Index) for task t_j (Eqn. (6.3))
$\mathbf{x}_i$	The position and velocity vector of agent a_i
$\mathcal{N}_i$	The neighbour agent set of agent a_i, given a communication network

$o_q \in \mathbb{R}^{n_d \times 1}$ indicates the central position vector of the q-th obstacle.

Each agent a_i has its (expected available) *work resource* (or *work capacity*) w_{im}, which varies depending on work position $p_m \in \mathcal{P}$. For example, given the agent's initial work resource (denoted by $w_{i,0}$), the cost to transition to work position p_m (denoted by c_{im}) may result in a different level of expected work capacity at last. This is also the case if the work position affects the agent's work efficiency (e.g., in the cooperative jamming mission [7], a position closer to a target radar provides higher jamming effectiveness). Considering this, work capacity w_{im} can be defined as

$$w_{im} := \eta_m(w_{i,0} - c_{im}), \tag{6.1}$$

where $\eta_m \in [0, 1]$ is the *work efficiency ratio* associated with position p_m.

For each task t_j to be completed, its *minimum (workload) requirement* R_j should be met by its *(task-specific) coalition*, denoted by $\mathcal{S}_j \subseteq \mathcal{A}$ (note that $\mathcal{S}_{j_1} \cap \mathcal{S}_{j_2} = \emptyset$ if $j_1 \neq j_2$). We regard that the task is completed if the aggregated work capacities of the coalition exceed the minimum requirement. Having extra agents besides those for marginally satisfying all the tasks' minimum requirements, it would be desirable to equitably distribute them as proportional to the requirements as possible to improve the system robustness.

Furthermore, there is a particular final time $\mathtt{t}_f$ by when all the agents should reach appropriate work positions without any collisions. Such a deadline may be an important factor for success in urgent missions, for instance, those in military applications or those in the search and rescue domain.

In such a prescribed mission environment, the robotic swarm's main decision-making issues may include:

1. To form task-specific coalitions and choose work positions in a way that (a) satisfies every task's minimum requirement at least and (b) equitably distributes the agents' work capacities in proportional to the tasks' requirements as possible;

2. To maximise the agents' work capacities by reducing travelling costs as well as exploiting positions with higher work efficiency; and

3. To generate shortest trajectories towards the selected positions, while avoiding any collisions with other agents or obstacles.

This problem as a whole is briefly illustrated in Figure 6.1 and can be formally defined as follows:

Problem 1 (*Original problem*)**.**

$$\max_{\{y_{im}\}} \big(\min_{\forall t_j \in \mathcal{T}} \alpha_j \big), \tag{6.2}$$

where

$$\alpha_j := \frac{\sum_{\forall a_i \in \mathcal{S}_j} \sum_{\forall p_m \in \mathcal{P}_j} w_{im} y_{im}}{R_j}, \tag{6.3}$$

subject to

$$\alpha_j \geq 1, \quad \forall t_j \in \mathcal{T}, \tag{6.4}$$

$$\sum_{\forall p_m \in \mathcal{P}} y_{im} \leq 1, \quad \forall a_i \in \mathcal{A}, \tag{6.5}$$

$$y_{im} \in \{0, 1\}, \quad \forall a_i \in \mathcal{A}, \forall p_m \in \mathcal{P}. \tag{6.6}$$

Here, α_j is the *requirement satisfaction index (RSI)* for task t_j, which should be equal to or greater than one to comply with the task's requirement (i.e., Equation (6.4)). y_{im} is a binary variable that is one if agent a_i is assigned to position p_m or zero otherwise. Equation (6.5) indicates that, because every agent has limited work resource, it can not execute more than two tasks in a mission and thus can be assigned to one working position at most. For fair allocation, the objective of this problem is to find an assignment set $\{y_{im}\}$ maximising the minimum value of α_j. Work capacity w_{im} is defined by Equation (6.1), where c_{im} is set by agent a_i's cost-to-go function $f_c^i(d_{im})$, which is an increasing function with regard to the corresponding travelling distance d_{im}, i.e.,

$$c_{im} := f_c^i(d_{im}). \tag{6.7}$$

The agent regards d_{im} as the length of the shortest collision-free trajectory towards position p_m as follows:

$$d_{im} := \min_{\mathbf{u}_i} \int_{\mathtt{t}_0}^{\mathtt{t}_f} ||G\dot{\mathbf{x}}_i(\mathtt{t})||\ d\mathtt{t}, \tag{6.8}$$

subject to

$$\dot{\mathbf{x}}_i(\mathtt{t}) = \mathbf{f}(\mathbf{x}_i(\mathtt{t})) + B\mathbf{u}_i(\mathtt{t}), \quad \forall \mathtt{t} \in [\mathtt{t}_0, \mathtt{t}_f], \tag{6.9}$$

$$\mathbf{x}_i(\mathsf{t}_0) = \mathbf{x}_{i,0}, \tag{6.10}$$

$$\mathbf{x}_i(\mathsf{t}_f) = [p_m; \mathbf{0}_{n_d \times 1}], \tag{6.11}$$

$$||G(\mathbf{x}_i(\mathsf{t}) - \mathbf{x}_l(\mathsf{t}))|| \geq r_{col},\ \forall \mathsf{t} \in [\mathsf{t}_0, \mathsf{t}_f], \forall a_l \in \mathcal{A} \setminus \{a_i\}, \tag{6.12}$$

$$||G\mathbf{x}_i(\mathsf{t}) - o_q|| \geq r_{obs,q}, \quad \forall \mathsf{t} \in [\mathsf{t}_0, \mathsf{t}_f], \forall o_q \in \mathcal{O}, \tag{6.13}$$

$$||\mathbf{u}_i(\mathsf{t})|| \leq U_{max}, \quad \forall \mathsf{t} \in [\mathsf{t}_0, \mathsf{t}_f], \tag{6.14}$$

$$||H\mathbf{x}_i(\mathsf{t})|| \leq V_{max}, \quad \forall \mathsf{t} \in [\mathsf{t}_0, \mathsf{t}_f], \tag{6.15}$$

where $\mathbf{x}_i(\mathsf{t}) = [\mathbf{p}_i(\mathsf{t}); \dot{\mathbf{p}}_i(\mathsf{t})]$; $\mathbf{p}_i(\mathsf{t}) \in \mathbb{R}^{n_d \times 1}$ and $\mathbf{u}_i(\mathsf{t}) \in \mathbb{R}^{n_d \times 1}$ are the agent's position vector and control vector at time t, respectively; $\mathbf{f}(.)$ represents its motion dynamics; and $\mathbf{x}_{i,0}$ is the given initial state at the initial time t_0. Here, $G = [I_{n_d \times n_d}\ \mathbf{0}_{n_d \times n_d}]$, $H = [\mathbf{0}_{n_d \times n_d}\ I_{n_d \times n_d}]$, $B = [\mathbf{0}_{n_d \times n_d}\ I_{n_d \times n_d}]^\top$. Each agent needs to keep the collision-avoidance distance r_{col} from other agents and $r_{obs,q}$ from obstacle o_q while moving to the assigned position (i.e., Equations (6.12)–(6.13)). Moreover, the collision-free trajectory should be feasible in terms of the agent's dynamics and physical constraints such as its available maximum control power U_{max} and velocity V_{max} (i.e., Equations (6.14)–(6.15)). □

Assumption 1 (*The number of agents*)**.** Given an instance of Problem 1, the number of the agents n_a is large enough such that there is at least a feasible agent-position assignment that satisfies every task's minimum workload requirement, i.e., $\alpha_j \geq 1$ for $\forall t_j \in \mathcal{T}$. Otherwise, there is no solution for the instance.

Assumption 2 (*Obstacle modelling*)**.** Each obstacle is enclosed with a circle (or sphere) with a radius of r_{ops}. Note that a complex-shaped obstacle can be addressed by introducing multiple circle-shaped obstacle avoidance areas.

Assumption 3 (*Robot capability*)**.** Each robot can know its position and follow the trajectory obtained from the proposed framework. The robot is also capable of hovering such as quadrotors.

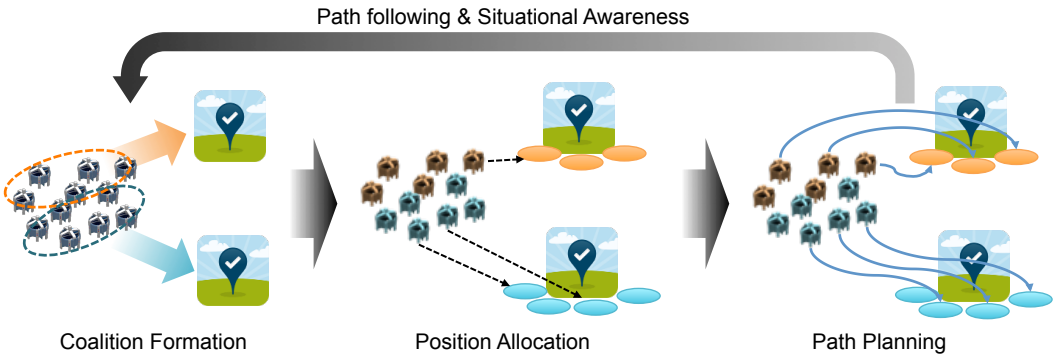


Figure 6.2: The proposed framework consists of three phases: coalition formation (Problem 2), position allocation (Problem 3), and path planning (Problem 4).

6.3 Decoupling to subproblems: coalition formation, position allocation, and path planning

Our idea is to split the original problem into three subproblems, i.e., *coalition formation*, *position allocation*, and *path planning* problems, and to make the agents address the subproblems sequentially and repeatedly along with a finite time-horizon, as illustrated in Figure 6.2, in a decentralised manner. In the first phase, the agents partition themselves into disjoint task-specific coalitions. Once an agreed set of coalitions $\{\mathcal{S}_j\}$ are determined, every agent in $\mathcal{S}_j$ selects a working position amongst all the available positions for task t_j, i.e., $p_m \in \mathcal{P}_j$. After that, all the agents generate and follow collision-free trajectories towards the selected working positions. The entire process is executed again before reaching the end of the time-horizon. In this section, we will show how the original problem can be decoupled into the three subproblems.

6.3.1 Coalition formation problem

Firstly, we introduce a decision variable $x_{ij} \in \{0, 1\}$, which equals to one if agent a_i joins coalition $\mathcal{S}_j$ and zero otherwise. It is implied from $x_{ij} = 1$ that the agent will be allocated to one of the positions for the task (i.e., $\forall p_m \in \mathcal{P}_j$). From this, it turns out

that $\sum_{\forall p_m \in \mathcal{P}_j} y_{im} = 1$. Accordingly, it can be equivalently said that

$$x_{ij} \equiv \sum_{\forall p_m \in \mathcal{P}_j} y_{im}. \tag{6.16}$$

We also introduce *abstracted cost* of agent a_i for task t_j, denoted by $\bar{c}_{ij}$, which abstracts all the costs towards the task's working positions, i.e., $\bar{c}_{ij} \approx c_{im}$ for $\forall p_m \in \mathcal{P}_j$. The abstracted cost is defined by

$$\bar{c}_{ij} := f_c^i(\bar{d}_{ij}), \tag{6.17}$$

where $\bar{d}_{ij}$ is the shortest travelling distance of agent a_i towards task t_j's central position (denoted by $p_{j,0}$) without consideration of the inter-agent collision avoidance constraints in (6.12). For clear explanation, Figure 6.3 illustrates the notations introduced.

Since the effectiveness of working positions will not be considered in this subproblem, for the moment we let $\eta_m = 1$. As we did for the cost, we also abstract the agent work resource in Equation (6.1) as *abstracted work resource* $\bar{w}_{ij} \approx w_{im}$ for $\forall p_m \in \mathcal{P}_j$. Thus, the equation can be rewritten as

$$\bar{w}_{ij} := w_{i,0} - \bar{c}_{ij}. \tag{6.18}$$

Under the aforementioned abstractions, α_j in (6.3) becomes equivalent to $\bar{\alpha}_j := \sum_{\forall a_i \in \mathcal{A}} \bar{w}_{ij} x_{ij} / R_j$, which can be derived by substituting w_{im} with $\bar{w}_{ij}$ and using Equation (6.16). Eventually, the original problem is reduced to the coalition formation problem defined as follows:

Problem 2 (*Coalition formation for fair resource allocation*)**.**

$$\max_{\{x_{ij}\}} \big(\min_{\forall t_j \in \mathcal{T}} \bar{\alpha}_j \big), \tag{6.19}$$

where

$$\bar{\alpha}_j := \frac{\sum_{\forall a_i \in \mathcal{A}} \bar{w}_{ij} x_{ij}}{R_j}, \tag{6.20}$$

subject to

$$\bar{\alpha}_j \geq 1, \quad \forall t_j \in \mathcal{T}, \tag{6.21}$$

$$\sum_{\forall t_j \in \mathcal{T}} x_{ij} \leq 1, \quad \forall a_i \in \mathcal{A}, \tag{6.22}$$

$$x_{ij} \in \{0,1\}, \quad \forall a_i \in \mathcal{A}, \forall t_j \in \mathcal{T}. \tag{6.23}$$

□

6.3.2 Position allocation problem

After forming coalitions, agents in each coalition $\mathcal{S}_j$ have to decide working positions amongst $\mathcal{P}_j$ to execute the assigned task t_j. In this phase, we consider the work efficiency of working positions η_m, which was not considered in the coalition formation problem. This agent-position allocation problem, which is the one-to-one assignment of independent single-position-required agents to independent single-agent-required positions, can be formulated by a linear assignment problem from the combinatorial optimisation literature [21]:

Problem 3 (*Position allocation*)**.** Given coalition $\mathcal{S}_j$ and a set of positions $\mathcal{P}_j$, the objective is to find an assignment such that

$$\max_{\{y_{im}\}} \sum_{\forall a_i \in \mathcal{S}_j} \sum_{\forall p_m \in \mathcal{P}_j} \tilde{w}_{im} y_{im} \quad \text{subject to} \tag{6.24}$$

$$\sum_{\forall p_m \in \mathcal{P}_j} y_{im} = 1, \quad \forall a_i \in \mathcal{S}_j, \tag{6.25}$$

$$\sum_{\forall a_i \in \mathcal{S}_j} y_{im} \leq 1, \quad \forall p_m \in \mathcal{P}_j, \tag{6.26}$$

$$y_{im} \in \{0,1\}, \quad \forall a_i \in \mathcal{S}_j, \forall p_m \in \mathcal{P}_j. \tag{6.27}$$

Here, $\tilde{w}_{im}$ is the expected available work resource of agent a_i when it arrives at position $p_m \in \mathcal{P}_j$ via the corresponding task's central position $p_{j,0}$, defined by

$$\tilde{w}_{im} := \eta_m(\bar{w}_{ij} - \tilde{c}_{im}), \tag{6.28}$$

where $\tilde{c}_{im} := f_c^i(||p_m - p_{j,0}||)$. □

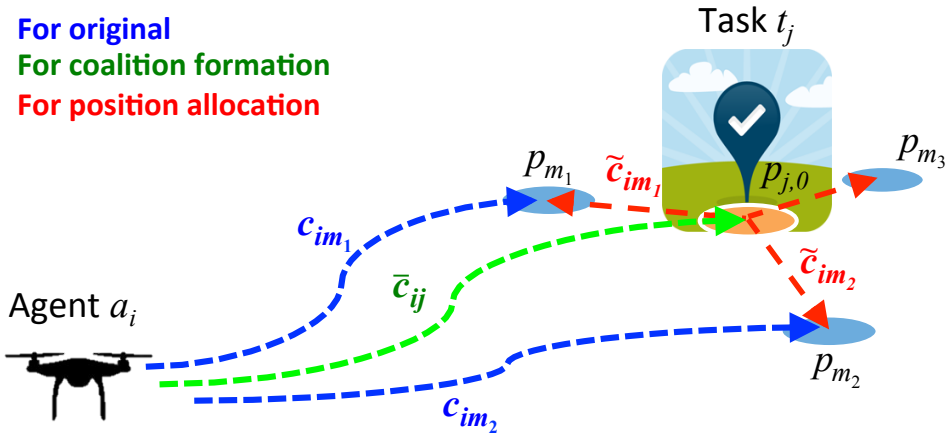


Figure 6.3: An illustration of c_{im}, $\bar{c}_{ij}$, and $\tilde{c}_{im}$. They are used for the original problem, the coalition formation subproblem, and the position allocation subproblem, respectively.

6.3.3 Path planning problem

In the last phase, given the position resulted by solving Problem 3, denoted by p_i^* for agent a_i, the agent needs to generate a collision-free trajectory that minimises its travelling distance, as follows:

Problem 4 (*Path planning with collision avoidance*)**.** For each agent $a_i \in \mathcal{A}$,

$$\min_{\mathbf{u}_i} \int_{\mathtt{t}_0}^{\mathtt{t}_f} ||G\dot{\mathbf{x}}_i(\mathtt{t})||\, d\mathtt{t}, \tag{6.29}$$

subject to (6.9), (6.10), (6.12)–(6.15), and

$$\mathbf{x}_i(\mathtt{t}_f) = [p_i^*; \mathbf{0}_{n_d \times 1}]. \tag{6.30}$$

□

6.3.4 Assumptions

The followings are assumptions considered in this study.

Assumption 4 (*Agents' network*)**.** The communication network of the agents is at least strongly connected as well as satisfies the property in Assumption 10, which will be described in Section 6.7. Given the network at time $\mathtt{t}$, $\mathcal{N}_i(\mathtt{t})$ denotes a set of neighbours nearby agent a_i, i.e., $\mathcal{N}_i(\mathtt{t}) = \{a_l \in \mathcal{A} \mid ||G(\mathbf{x}_i(\mathtt{t}) - \mathbf{x}_l(\mathtt{t}))|| \leq R_{comm}\}$, where R_{comm} is the agent's communication radius.

Assumption 5 (*Accessible information*)**.** Every agent a_i can sense every neighbour agent $a_l \in \mathcal{N}_i(\mathtt{t})$, and obtain its local information such as $\bar{c}_{lj}$, $\bar{w}_{lj}$, $\mathbf{x}_l$ within a reasonably short time. The agent also knows the mission scenario information regarding tasks, working positions and obstacles, e.g., R_j, $\mathcal{P}$, η_m, $\mathcal{O}$.

6.4 Coalition Formation

6.4.1 Algorithm

By letting $\bar{\alpha}_{\min} := \min_{\forall t_j \in \mathcal{T}} \bar{\alpha}_j$ (called *the minimum RSI*), it can be said that the objective of Problem 2 is to find the maximised $\bar{\alpha}_{\min}$, denoted by $\bar{\alpha}_{\max\min}$. Supposing that we already know the value, the corresponding optimal assignment $\{x_{ij}\}$ complies with the following conditions:

Condition 1. There is no unused agent. All the given agents join to any coalition (i.e., $\sum_{\forall t_j \in \mathcal{T}} x_{ij} = 1$, $\forall a_i \in \mathcal{A}$).

Condition 2. Given $\bar{\alpha}_{\max\min}$, it holds that

$$\bar{\alpha}_j \geq \bar{\alpha}_{\max\min} \quad \forall t_j \in \mathcal{T}.$$

Condition 3. Within constraints (6.22)–(6.23) and Condition 2, the total work capacities of the agents (i.e., $\sum_{\forall a_i \in \mathcal{A}} \sum_{\forall t_j \in \mathcal{T}} \bar{w}_{ij} x_{ij}$) are nearly maximised. Equivalently, from the definition of $\bar{w}_{ij}$ in (6.18), it can be also said that the total cost of the agents (i.e., $\sum_{\forall a_i \in \mathcal{A}} \sum_{\forall t_j \in \mathcal{T}} \bar{c}_{ij} x_{ij}$) is almost minimised. Note that the optimal assignment $\{x_{ij}\}$ for the max-min optimisation is not necessarily optimal for maximisation of the total work capacities. However, the extent of the degradation is expected to be not significant due to Condition 2 and the supposition that $\bar{\alpha}_{\max\min}$ was already optimised.

Our fundamental approach towards Problem 2 is to search the optimised $\bar{\alpha}_{\max\min}$ and its corresponding assignment set that meets all the conditions. Assuming that a certain value of $\bar{\alpha}_{\max\min} \geq 1$ is given as the nominal value, we will firstly find an assignment that satisfies Condition 2 while minimising the total cost of the agents (i.e., Conditions 3). This subproblem can be formulated as *MinGAP-MR (Minimisation version of General Assignment Problem with Minimum Requirements)* [22], which is defined as follows:

Problem 5 (*MinGAP-MR*)**.** Given a nominal value of $\bar{\alpha}_{\max\min}$,

$$\min_{\{x_{ij}\}} \quad \sum_{\forall a_i \in \mathcal{A}} \sum_{\forall t_j \in \mathcal{T}} \bar{c}_{ij} x_{ij} \tag{6.31}$$

subject to (6.22), (6.23), and for $\forall t_j \in \mathcal{T}$

$$\sum_{\forall a_i \in \mathcal{A}} \bar{w}_{ij} x_{ij} \geq \bar{\alpha}_{\max\min} R_j. \tag{6.32}$$

□

Note that Condition 2 can be transformed to Equation (6.32) by the definition of $\bar{\alpha}_j$, i.e., Equation (6.20). If the resultant assignment also conforms with Condition 1, the nominal value is a suboptimal objective function value of Problem 2. If either of Conditions 1 or 2 fails, the nominal value $\bar{\alpha}_{\max\min}$ needs to be changed, and this process is iteratively executed until both conditions are fulfilled.

Of importance is how to search $\bar{\alpha}_{\max\min}$ properly and quickly. To this end, we adopt the concept of binary search [23]. Firstly, we set the nominal value of $\bar{\alpha}_{\max\min}$ as its possible maximum value J_{LHR}, which is defined in the following proposition.

Proposition 1. *The optimal objective function value of Problem 2 is upper bounded by its* Linear-and-Homogeneous Relaxation *(LHR) objective function value, which is defined by*

$$J_{LHR} := \frac{\sum_{\forall a_i \in \mathcal{A}} (\max_{\forall t_j \in \mathcal{T}} \bar{w}_{ij})}{\sum_{\forall t_j \in \mathcal{T}} R_j}. \tag{6.33}$$

Proof. Assume that the binary constraint (6.23) is relaxed so that an agent's resource can be assigned fractionally (i.e. *Linear relaxation*, $x_{ij} \in [0, 1]$). The optimal solution

in the relaxed problem is such that the maximised value of $\sum_{\forall a_i \in \mathcal{A}} \sum_{\forall t_j \in \mathcal{T}} \bar{w}_{ij} x_{ij}$ is shared to the given tasks in exactly proportion to their minimum requirements. This shared value always upper bounds the optimal objective function of Problem 2. The maximised value of $\sum_{\forall a_i \in \mathcal{A}} \sum_{\forall t_j \in \mathcal{T}} \bar{w}_{ij} x_{ij}$ is at most $\sum_{\forall a_i \in \mathcal{A}} (\max_{\forall t_j \in \mathcal{T}} \bar{w}_{ij})$, where each agent is assumed to have its maximum resource regardless of the tasks (i.e., *(Maximum) Homogeneous relaxation*). Combining the two relaxations yields the value as shown in Equation (6.33), which also upper bounds the optimal objective function of Problem 2. □

Algorithm 1 Coalition formation with fair allocation

1: $\bar{\alpha}_{\max\min}$:= the value of Equation (6.33)
2: $\beta := \bar{\alpha}_{\max\min}$
3: **repeat**
4: $\quad \beta \leftarrow \beta/2$
5: $\quad \{x_{ij}\}$:= the solution for Problem 5 (by Algorithm 2)
6: $\quad$ Modify $\bar{\alpha}_{\max\min}$ according to Table 6.2
7: **until** Conditions 1 and 2 are met as well as $\beta \leq \epsilon_{CF}$
8: **return** $\{x_{ij}\}$

The proposed coalition formation process is described in Algorithm 1. After setting the nominal value $\bar{\alpha}_{\max\min} = J_{LHR}$ intially, we try to find a solution assignment for Problem 5 by Algorithm 2, which will be shown in the following section. Depending on the solution's fulfillness with regard to Conditions 1 and 2, the nominal value is modified as shown in Table 6.2. In the table, β is the amount of variation to adjust the previous nominal value. β is initially set by the half of J_{LHR}, and is reduced to one half at every search.

Failure of Condition 2 means that there exists no solution for Problem 5 with the given nominal value $\bar{\alpha}_{\max\min}$. Therefore, the value needs to be reduced. Failure of Condition 1 implies that there must be another assignment that yields a higher value of $\bar{\alpha}_{\max\min}$ exploiting all the given agents. This may be the case even if both conditions are fulfilled, and hence, we try to increase the nominal value. This iterative search is executed until the time not only when Conditions 1 and 2 are simultaneously satisfied but also when β is less than a certain bound ϵ_{CF}. Note that the subroutine algorithm

Table 6.2: How to change the nominal value of $\bar{\alpha}_{\max\min}$ in Algorithm 1

Given $\{x_{ij}\}$	Condition 2 fulfilled	Condition 2 failed
Condition 1 fulfilled	Increase $\bar{\alpha}_{\max\min}$	Decrease $\bar{\alpha}_{\max\min}$
Condition 1 failed	Increase $\bar{\alpha}_{\max\min}$	Not applicable

· When increasing: $\bar{\alpha}_{\max\min} \leftarrow \bar{\alpha}_{\max\min} + \beta$

· When decreasing: $\bar{\alpha}_{\max\min} \leftarrow \bar{\alpha}_{\max\min} - \beta$

for Problem 5 (i.e., Algorithm 2) does not provide the case where both conditions fail, the details of which will be described in Remark 1 in the following section.

6.4.2 The subroutine for MinGAP-MR

This section addresses Problem 5 by using the distributed game-theoretical algorithm proposed in our previous study [22], where the problem is modelled as a coalition formation game of selfish agents who have different preferences regarding task-specific coalitions. We introduce some necessary definitions and notations for the algorithm. We define a *partition* as a set $\Pi = \{\mathcal{S}_1, ..., \mathcal{S}_{n_t}, \mathcal{S}_0\}$ that disjointly partitions the agent set $\mathcal{A}$. Here, $\mathcal{S}_0$ is the coalition of agents who do "not work any task" in $\mathcal{T}$. Given the partition, $\Pi(i)$ indicates the index of the coalition to which agent a_i joins. Let e_{ij} denote the *expense* for agent a_i with regard to coalition $\mathcal{S}_j$ (or equivalently task t_j). Under this algorithm, every agent tends to select the coalition requiring the lowest expense.

The objective of the algorithm is to find a *Nash stable* partition, where no agent will unilaterally deviate, which is defined as follows:

Definition 1 (*Nash stable partition*)**.** A partition Π is said to be *Nash stable* if it holds that, for every agent $a_i \in \mathcal{A}$, $e_{i\Pi(i)} \leq e_{ij}$, $\forall \mathcal{S}_j \in \Pi$.

Given Π, the expense e_{ij} is defined as follows:

$$e_{ij} := \begin{cases} \bar{c}_{ij+} & \text{if } a_i \in \hat{\mathcal{S}}_j, \\ \infty & \text{otherwise,} \end{cases} \tag{6.34}$$

where $\bar{c}_{ij+}$ is the *learnt cost* of agent a_i for task t_j (for now, we set $\bar{c}_{ij+}$ to $\bar{c}_{ij}$); and $\hat{\mathcal{S}}_j$ is the set of agents eligible to join the coalition for task t_j. Agent a_i can ascertain its *eligibility* by an algorithm for MinKP (Minimum Knapsack Problem): $\hat{\mathcal{S}}_j := \textsc{MinKP}(\mathcal{S}_j \cup \{a_i\}, \bar{\alpha}_{\max\min} R_j, \mathcal{W}_j, \mathcal{C}_j)$[1], where $\mathcal{S}_j \in \Pi$, $\mathcal{W}_j = \{\bar{w}_{kj} | \forall k : a_k \in \mathcal{S}_j \cup \{a_i\}\}$ and $\mathcal{C}_j = \{\bar{c}_{kj} | \forall k : a_k \in \mathcal{S}_j \setminus \{a_i\}\} \cup \{\bar{c}_{ij+}\}$.

Since there may exist any possible divergence from a Nash stable partition, we let each agent a_i update its learnt costs as follows:

$$\bar{c}_{ij+} \leftarrow \bar{c}_{ij+} + \lambda \cdot \bar{c}_{ij}, \tag{6.35}$$

where $\lambda > 0$ is the *learning rate*, and initially $\bar{c}_{ij+} = \bar{c}_{ij}$. Whenever agent a_i changes its decision from coalition $\mathcal{S}_j$ to another one, the agent learns that the previous choice was not suitable for itself and penalises it gradually by the learning rate as Equation (6.35). This can be called as *tabu learning* [25]. This iterative decision-making and learning process guarantees that a Nash stable partition always can be determined [22].

Its detailed procedure is described in Algorithm 2. For every agent a_i, given the current locally-known partition information, denoted by Π^i, if the agent does not satisfy with the partition (Line 5), then it investigates every coalition $\mathcal{S}_j \in \Pi^i$ and computes the corresponding expense e_{ij} according to Equation 6.34 (Lines 6–9). If the least expense value provides infinity, then the agent set the preferred coalition index j^* as zero (Lines 11–13), meaning that it will move to $\mathcal{S}_0$. If the most preferred coalition is not the existing one, the agent updates the learnt cost for the existing coalition, joins to $\mathcal{S}_{j^*}$, amends Π^i to reflect its new decision, increases r^i (which is the number of evolutions of the partition), and generates a new random time stamp s^i (Line 14–20). Then, the agent constructs a message $\mathsf{msg}^i := \{r^i, s^i, \Pi^i, \mathsf{satisfied}^i\}$ and sends it to other neighbour agents, and vice versa (Line 23). Amongst the multiple partition information, only one can be distributedly chosen by any agent at last through the distributed mutex

[1]Please refer to Definition 2 in Appendix B for more details. This study uses MinGreedy algorithm [24] as MinKP.

algorithm [26] in Appendix C (i.e., Algorithm 5), denoted by D-MUTEX (Line 25). The distributed mutex algorithm enables Algorithm 2 to be executable asynchronously and distributedly as long as the network of the given agents is at least strongly-connected.

Remark 1. Given an outcome $\{x_{ij}\}$ from Algorithm 2, the simultaneous failure of Conditions 1 and 2 does not happen. According to Lines 6–13, if there exists any task t_j not satisfying the constraint (6.32) (i.e., Condition 2 fails), there must be no agent who wants to join $\mathcal{S}_0$. This means that all the agents join to any task-specific coalitions (i.e., Condition 1 is fulfilled).

Remark 2 (*How to reduce unnecessary process in Algorithm* 1)**.** We set that if agent a_i find the index of its most preferred coalition as $j^* = 0$ (Line 12), the agent broadcasts a notification signal to other agents so that they notice Condition 1 failed. This setting enables the agents to early stop Algorithm 2 and change the nominal value of $\bar{\alpha}_{\max\min}$ (as Table 6.2), and thus reduces possible unnecessary computation.

6.4.3 Analysis

We now discuss the properties of the proposed coalition formation algorithm.

Proposition 2 (*Convergence of Algorithm 1*)**.** *Algorithm 1 terminates in a finite number of iterations.*

Proof. For the coalition formation algorithm to terminate, Line 7 in Algorithm 1 should be met. Since β is gradually reduced by one half, it is obvious that $\beta \leq \epsilon_{CF}$ within a finite number of iterations. Now let us investigate if there will always be a moment when Conditions 1 and 2 are simultaneously fulfilled. Considering the properties of both conditions, it can be said that (1) Condition 1 is met if the nominal value of $\bar{\alpha}_{\max\min}$ is within $[\bar{\alpha}_{\mathrm{c}_1}, \infty]$, where $\bar{\alpha}_{\mathrm{c}_1} < \infty$; (2) Condition 2 is satisfied if the nominal value of $\bar{\alpha}_{\max\min}$ is within $[0, \bar{\alpha}_{\mathrm{c}_2}]$, where $\bar{\alpha}_{\mathrm{c}_2} > 0$. From this, if $\bar{\alpha}_{\mathrm{c}_2} < \bar{\alpha}_{\mathrm{c}_1}$, then there must be a possibility that Conditions 1 and 2 simultaneously fail. However, as described in Remark 1, this is not the case, and hence $[\bar{\alpha}_{\mathrm{c}_1}, \bar{\alpha}_{\mathrm{c}_2}]$, which is the range when both conditions are fulfilled, always exists and is non-empty. It follows that the modification procedure of the nominal value $\bar{\alpha}_{\max\min}$ shown in Table 6.2 eventually makes the value

Algorithm 2 MinGAP-MR algorithm for each agent a_i

// Note: $\alpha_{\max\min}$ is given from Algorithm 1

// Initially, Π^i := the initial partition; $r^i \leftarrow 0$; $s^i \leftarrow 0$; $\mathsf{satisfied}^i \leftarrow \mathbf{0}_{n_a \times 1}$

if unexpected change of Π^i is sensed **then** *// For dynamic environments*

  $r^i \leftarrow r^i + 1$; $s^i \leftarrow 0$; $\mathsf{satisfied}^i \leftarrow \mathbf{0}_{n_a \times 1}$

end if

// Decision-making process begins

while $\mathsf{satisfied}^i \neq \mathbf{1}_{n_a \times 1}$ **do**

// Make a new decision if necessary

  if $\mathsf{satisfied}^i[i] = 0$ **then**

    for each $\mathcal{S}_j \in \Pi^i \setminus \{\mathcal{S}_0\}$ **do**

      $\hat{\mathcal{S}}_j := \textsc{MinKP}(\mathcal{S}_j \cup \{a_i\}, \bar{\alpha}_{\max\min} R_j, \mathcal{W}_j, \mathcal{C}_j)$

      Compute e_{ij} using Eqn. (6.34)

    end for

    $j^* = \text{argmin}_{\forall j} e_{ij}$; $e^* = \min_{\forall j} e_{ij}$

    if $e^* = \infty$ **then** *// No coalition is preferred*

      $j^* := 0$

    end if

    if $j^* \neq \Pi^i(i)$ **then**

      $\bar{c}_{i,\Pi^i(i)+} = \bar{c}_{i,\Pi^i(i)+} + \lambda \cdot \bar{c}_{i,\Pi^i(i)}$

      Join $\mathcal{S}_{j*}$ and update Π^i

      $r^i \leftarrow r^i + 1$

      $s^i \in \text{unif}[0, 1]$

      $\mathsf{satisfied}^i \leftarrow \mathbf{0}_{n_a \times 1}$

    end if

    $\mathsf{satisfied}^i[i] = 1$

  end if

// Broadcast the local information to neighbour agents

  Broadcast $\mathsf{msg}^i = \{r^i, s^i, \Pi^i, \mathsf{satisfied}^i\}$ and receive msg^l from its neighbours $\forall a_l \in \mathcal{N}_i$

// Select the valid partition from all the received messages

  Collect all the messages $\mathcal{M}^i_{rcv} = \{\mathsf{msg}^i, \forall \mathsf{msg}^l\}$

  $\{r^i, s^i, \Pi^i, \mathsf{satisfied}^i\} := \text{D-}\textsc{Mutex}(\mathcal{M}^i_{rcv})$

end while

return $\{x_{ij}\}$ corresponding to Π^i

converge to the range. The subroutine in Line 5 (i.e., Algorithm 2) terminates within a finite time [22], so does Algorithm 1. This completes the proof. □

Proposition 3 (*The number of iterations in Algorithm 1*)**.** *Suppose that, after κ iterations of the main loop of Algorithm 1 (i.e., Lines 3–7), the nominal value $\bar{\alpha}_{\max\min}$ remains the range where Conditions 1 and 2 are both fulfilled. Then, the maximum number of iterations happened is upper bounded by* $\max\{\kappa, \lceil \log_2 (J_{LHR}/\epsilon_{CF}) \rceil\}$.

Proof. As β reduces to its half after each iteration, it can be said that $\beta = J_{LHR} \cdot (\frac{1}{2})^n$, where n is the number of iterations happened. Until the time when $\beta \leq \epsilon_{CF}$, the required number of iterations is $\lceil \log_2 (J_{LHR}/\epsilon_{CF}) \rceil$. For Algorithm 1 to finish, Conditions 1 and 2 additionally need to be satisfied. Hence, the maximum number of iterations is upper bounded by $\max\{\kappa, \lceil \log_2 (J_{LHR}/\epsilon_{CF}) \rceil\}$. □

6.5 Position Allocation

After the coalition formation process, agents in each coalition $\mathcal{S}_j$ have to choose working positions amongst $\forall p_m \in \mathcal{P}_j$ to execute the assigned task t_j (i.e., Problem 3). Since this problem can fall into ST-SR case in multi-robot task allocation categories [8], it could be optimally solved by a linear programming (e.g., Kuhn's Hungarian method [27]) in $O(|\mathcal{S}_j||\mathcal{P}_j|^2)$ time. Note that any imbalance regarding the number of agents and positions can be addressed by including dummy agents or positions as required [9]. Please refer to the review papers [8, 9] for more details.

This section addresses Problem 3 in a more computationally efficient manner under the following assumptions:

Assumption 6 (*Initial positions of agents from tasks*)**.** When a position allocation process begins, the distance from agent $a_i \in \mathcal{S}_j$ to task t_j's central position $p_{j,0}$ is much larger than the spatial difference from any position $p_m \in \mathcal{P}_j$ to $p_{j,0}$. Hence, the costs of the agent to transition to any positions in $\mathcal{P}_j$ are approximately the same as that to $p_{j,0}$, i.e., $c_{im} \approx \bar{c}_{ij}$, $\forall p_m \in \mathcal{P}_j$. In other words, the difference between $p_{j,0}$ and each $p_m \in \mathcal{P}_j$ can be negligible, and thereby it follows that $\tilde{c}_{im} \approx 0$ in Equation (6.28). Hence, it can be approximated as

$$\tilde{w}_{im} := \eta_m(\bar{w}_{ij} - \tilde{c}_{im}) \approx \eta_m \bar{w}_{ij} \tag{6.36}$$

Assumption 7 (*Work efficiency depending on proximity*)**.** Given any two positions $p_{m_1}, p_{m_2} \in \mathcal{P}_j$, if p_{m_1} is closer to task t_j than p_{m_2}, then an agent at p_{m_1} works more efficiently than one at p_{m_2}, i.e., $\eta_{m_1} > \eta_{m_2}$. Likewise, $\eta_{m_1} = \eta_{m_2}$ if p_{m_1} and p_{m_2} are identically distant from the task. This assumption can be considered reasonable because, for example, in cooperative jamming mission [7] or in vision surveillence mission, an agent's proximity brings stronger work efficiency (e.g., jamming effectiveness or sensing capability).

Proposition 4. *Under Assumptions 6 and 7, the optimal solution for Problem 3 is the allocation resulted by assigning higher work-capacity agents into higher work-efficiency positions.*

Proof. Since Problem 3 is only related to each task t_j, we firstly let $\bar{w}_i := \bar{w}_{ij}$ for simplicity. Without loss of generality, it can be said that there are a set of agents $\mathcal{S}_j$ such that $\bar{w}_1 \geq \bar{w}_2 \geq ... \geq \bar{w}_{|\mathcal{S}_j|}$, and $|\mathcal{S}_j|$ of working positions such that $\eta_1 \geq \eta_2 \geq ... \geq \eta_{|\mathcal{S}_j|}$.

For any two numbers $m, n \in \{1, 2, ..., |\mathcal{S}_j|\}$ such that $m \leq n$,

$$\eta_m \bar{w}_m + \eta_n \bar{w}_n \geq \eta_m \bar{w}_n + \eta_n \bar{w}_m. \tag{6.37}$$

This is because, considering that $\delta := \eta_m - \eta_n \geq 0$, the left-hand side in (6.37) becomes $(\eta_n + \delta)\bar{w}_m + (\eta_m - \delta)\bar{w}_n = \eta_n \bar{w}_m + \eta_m \bar{w}_n + \delta(\bar{w}_m - \bar{w}_n)$, and hence, this is always greater than or equal to the right-hand side in (6.37).

From this, it is obvious that, given any two agents in $\mathcal{S}_j$, it is always better to assign the higher work-resource agent to the higher work-efficiency position. According to Assumption 7, such a position is the one closer to task t_j. This also holds for multiple agents more than two. Hence, by assigning higher work-capacity agents into higher work-efficiency positions, we can find the optimal assignment for Problem 3 under Assumptions 6 and 7. □

Owing to Proposition 4, a simple sorting algorithm can be utilised to find the optimal solution for the approximated Problem 3. There exist various types of sorting algorithms such as Quicksort, Mergesort, and Heapsort, and their complexities are averagely known as $O(n \log n)$, where n is the number of the given items to be sorted [28]. Since such a sorting algorithm is too simple, in this position allocation process, every agent executes the algorithm in parallel.

6.6 Path Planning with Collision Avoidance

After the position allocation process in the previous section, the agents need to generate collision-free trajectories in a way that minimises their travelling distances (i.e., Problem 4). This chapter exploits the MPC-SCP path planning algorithm [20], which consists of a MPC loop and a SCP loop. Given a non-convex trajectory optimisation problem along with a nominal trajectory, the SCP loop recursively solves its approximated problem, which is convexified by the nominal trajectory as explained in the following paragraph (i.e., Problem 6), until the approximated solution is very close to the nominal one. In order to reduce the problem size, each agent considers inter-agent collision avoidance only for a time horizon from the current moment. While the myopic collision-free trajectory is being used for the time horizon, the next one is calculated before reaching the end of the horizon, which is the MPC loop. It is worth noting that the MPC-SCP algorithm can consider not only inter-agent collision avoidance but also obstacle avoidance, and that its real-time implementability was experimentally validated by using multiple quadrotors [19]. In this section, we briefly introduce how to utilise the MPC-SCP to Problem 4, then show the main difference of the proposed algorithm from the existing MPC-SCP.

Given the nominal trajectory of agent a_i (denoted by $\bar{\mathbf{x}}_i$), we firstly linearise and discretise the problem as described in Appendix A. In addition, the collision-avoidance constraints in (6.55) and (6.56) should be convex-approximated into the following affine constraints for a time horizon T_H: for every agent $a_i \in \mathcal{A}$,

$$\begin{aligned}(\bar{\mathbf{x}}_i[k] - \bar{\mathbf{x}}_l[k])^\top G^\top G(\mathbf{x}_i[k] - \bar{\mathbf{x}}_l[k]) \geq r_{col}||G(\bar{\mathbf{x}}_i[k] - \bar{\mathbf{x}}_l[k])||,\\ k = k_0, ..., \min\{T, k_0 + T_H\},\ \forall a_l \in \mathcal{N}_i[k_0] \cap \mathcal{I}_i,\end{aligned} \tag{6.38}$$

$$\begin{aligned}(G\bar{\mathbf{x}}_i[k] - o_q)^\top (G\mathbf{x}_i[k] - o_q) \geq r_{obs,q}||G\bar{\mathbf{x}}_i[k] - o_q||,\\ k = k_0, ..., \min\{T, k_0 + T_H\},\ \forall o_q \in \mathcal{O},\end{aligned} \tag{6.39}$$

where $\mathcal{N}_i[k]$ is the discretised-version notation of $\mathcal{N}_i(\mathtt{t}_k)$, and $\mathcal{I}_i \subseteq \mathcal{A}$ is the set of neighbour agents who are more important than agent a_i meaning that agent a_i should avoid them (the formal definition of $\mathcal{I}_i$ will be shown later in Assumption 8). The convexified

constraints (6.38) and (6.39) are sufficient conditions for the original collision-avoidance constraints (6.55) and (6.56) to hold [20]. Hence, Problem 4 can be reduced to the following convex-approximated problem:

Problem 6 (*Convex program for path planning*)**.** For each agent $a_i \in \mathcal{A}$, given nominal trajectories $\bar{\mathbf{x}}_l$ for $\forall a_l \in (\mathcal{N}_i[k_0] \cap \mathcal{I}_i) \cup \{a_i\}$ at the current discretised time step k_0,

$$\min_{\mathbf{x}_i} \sum_{k=k_0}^{T-1} ||G(\mathbf{x}_i[k+1] - \mathbf{x}_i[k])|| \tag{6.40}$$

subject to (6.52),(6.30),(6.57),(6.58),(6.38),(6.39). □

Assumption 8 (*Importances of agents* [19])**.** Let each agent a_i has its importance index ρ_i such that $\rho_i \neq \rho_l$ for $\forall l \neq i$. For the convergence of Algorithm 3, it should be that

$$\mathcal{I}_i = \{a_l \in \mathcal{N}_i[k_0] \mid \rho_l < \rho_i\}.$$

The SCP loop for path planning is described in Algorithm 3. Firstly, each agent a_i sets its nominal trajectory $\bar{\mathbf{x}}_i$ from the solution for Problem 6 without consideration of the inter-agent coalition avoidance constraint (Line 1). The agent shares the information with all the neighbours (Line 2). Let $\mathcal{K}_i$ denote a local variable indicating neighbour agents who have not found coalition-free trajectories yet. If this is the case for agent a_i, it tries to obtain the solution for Problem 6 (Line 6). If it is feasible, then the nominal trajectory is updated. Otherwise, the previous nominal trajectory $\bar{\mathbf{x}}_{i,prev}$ is set to any trajectory such that $||\bar{\mathbf{x}}_i[k] - \bar{\mathbf{x}}_{i,prev}[k]|| \not\leq \epsilon_{\text{SCP}}, \forall k$ (Lines 7–12). After sharing the information with all the neighbours (Line 14), agent a_i checks if the new trajectory is close enough to the previous nominal trajectory and collision-free (Lines 15-19), and updates $\mathcal{K}_i$ (Lines 20-21).

Remark 3 (*How to avoid infeasible solutions* in Line 6)**.** Although Problem 6 is a convex programming, there might be no feasible solution if the nominal trajectory $\bar{\mathbf{x}}_i$ is not proper. The convexified constraints (6.38) and (6.39) based on the nominal trajectory occasionally induce the event that some agents can not generate feasible trajectories using the given maximum velocity and control powers. In this case, it is a remedy to solve Problem 6 with temporarily-reduced r_{col} or $r_{obs,q}$ and then gradually increase them and solve the problem again. Alternatively, making the initial velocity profiles of the nominal trajectory become slowly also often works.

Algorithm 3 Collision-free path planning for each agent a_i

$\bar{\mathbf{x}}_i[k] :=$ the solution to Problem 6 without (6.38), $\forall k$
Communicate $\bar{\mathbf{x}}_i$ to all neighbour agents $a_l \in \mathcal{N}_i[k_0]$
$\mathcal{K}_i := \mathcal{N}_i[k_0] \cup \{a_i\}$
while $\mathcal{K}_i \neq \emptyset$ **do**
    if $a_i \in \mathcal{K}_i$ **then**
        $\mathbf{x}_i[k] :=$ the solution to Problem 6, $\forall k$
        if $\mathbf{x}_i[k]$ is feasible **then**
            $\bar{\mathbf{x}}_{i,prev}[k] := \bar{\mathbf{x}}_i[k]$, $\forall k$
            $\bar{\mathbf{x}}_i[k] := \mathbf{x}_i[k]$, $\forall k$
        else
            $\bar{\mathbf{x}}_{i,prev}[k] := \bar{\mathbf{x}}_i[k] + \epsilon_{\text{SCP}} \cdot \mathbf{1}_{2n_d \times 1}$, $\forall k$
        end if
    end if
    Communicate $\bar{\mathbf{x}}_i$ to all neighbours $a_l \in \mathcal{N}_i[k_0]$
    if $||\bar{\mathbf{x}}_i[k] - \bar{\mathbf{x}}_{i,prev}[k]|| < \epsilon_{\text{SCP}}$
     and $||G(\bar{\mathbf{x}}_i[k] - \bar{\mathbf{x}}_l[k])|| > r_{col}$ $\forall a_l \in \mathcal{N}_i[k_0] \cap (\mathcal{K}_i^{\mathsf{c}} \cup \mathcal{I}_i)$
     and $||G\bar{\mathbf{x}}_i[k] - \mathbf{o}_q|| > r_{obs,q}$ $\forall o_q \in \mathcal{O}$, $\forall k$, **then**
        Remove a_i from $\mathcal{K}_i$
    end if
    Communicate $\mathcal{K}_i$ to all neighbours $a_l \in \mathcal{N}_i[k_0]$
    $\mathcal{K}_i := \mathcal{K}_i \cap (\cup_{\forall l \in \mathcal{N}_i[k_0]} \mathcal{K}_l)$
end while
$\bar{\mathbf{x}}_i$ is the approximate solution to Problem 4

The main difference of Algorithm 3 from the existing work [20] is Lines 7–12: agents who are eventually not able to find feasible solutions for Problem 6 keep their previous trajectories, whereas other agents use new ones. When implementing the existing algorithm in the way it is (i.e., without the lines), agents who already obtained feasible trajectories have to wait to receive their neighbours' trajectories before proceeding Line 15. Although infeasible solutions can be avoided by Remark 3, this additional process might induce unnecessary waiting times for other agents in a synchronous process. Therefore, Algorithm 3 is designed such that some agents may temporarily skip their trajectory computations, while keeping the existing trajectories. We now show that, despite so, the proposed algorithm can converge to collision-free trajectories.

Proposition 5 (*Convergence of the path planning algorithm*)**.** *Even if some agents are temporarily not able to find feasible solutions to Problem 6, Algorithm 3 can terminate within a finite time and provide conflict-free trajectories for all the agents.*

Proof. According to Lines 7-12 in Algorithm 3, if agent a_i can not temporarily find a feasible solution to Problem 6, then its neighbour agents maintain $\bar{\mathbf{x}}_i$ as they knew before. For any agent amongst those neighbours, if it obtains a feasible solution in Line 6, the solution is conflict-free with regard to $\bar{\mathbf{x}}_i$. As such, asynchronous computation for new trajectories is more favourable for an agent to comply with the condition in Line 16, compared with the circumstance where all the agents obtain new solutions in Line 6. This is because each feasible solution $\mathbf{x}_i$ considers the corresponding neighbours' previous trajectories as the nominal values (as shown in Equation (6.38)), and thus the simultaneous update of every $\mathbf{x}_i$ $\forall a_i$ probably causes additional iterations. However, despite the simultaneous update, Morgan et al. [19] shows that all the agents can converge to collision-free trajectories within a finite time. Therefore, it can be said that the occurrence of temporary infeasible solutions in Line 6 does not hinder the convergence of Algorithm 3 towards conflict-free trajectories for all the agents. □

Moreover, we experimentally observed that this asynchronous process results in a faster agreement of the agents than a synchronous process, as pointed out in the work [29].

6.7 The Proposed Integrated Framework

This section presents our integrated framework including all the algorithms introduced previously (as shown in Algorithm 4). At the current time k_0, each agent firstly executes the coalition formation process after obtaining its abstracted costs (Lines 3–6). Note that $\bar{d}_{ij}$ (i.e., the shortest travelling distance of agent a_i towards task t_j's central position) can be geometrically obtained under Assumption 2. Then, the agent proceeds the position allocation process (Lines 7–11), and the path planning algorithm (Line 12) over the time horizon T_H. Although Algorithm 4 implies that all the subprocesses have the same update rates, in practice they may be different. For example, those for the coalition formation and the position allocation may be able to be executed in a less frequent manner than those for the path planning and following.

Since inter-agent collision avoidance is considered during the time horizon only, we need the following assumptions:

Assumption 9 (*Computational feasibility*)**.** Every agent's computational and communicational capability is such that the running time of each loop of Algorithm 4 (Lines 3–12) is less than $T_H\Delta\mathsf{t}$. This guarantees that there will be no inter-agent collision during the entire mission period.

Assumption 10 (*Detectable collisions* [20])**.** To guarantee that there is no unexpected collision with other neighbour agents during the time horizon T_H, R_{comm} and V_{max} should be such that

$$R_{comm} \geq 2V_{\max}T_H\Delta\mathsf{t} + r_{col} \quad \text{and} \quad \Delta\mathsf{t}V_{max} < r_{col}.$$

We conjecture that the integrated framework can be executed even by asynchronous behaviours of the agents to some extent. Amongst the subroutines comprising the framework, the path planning subroutine (i.e., Algorithm 3) needs a relatively more ideal environment: it requires *Local-Information Consistency Assumption (LICA)* (i.e., local information needs to be consistently known by a local agent group at each iteration) [30] over neighbour agents under a communication network holding Assumptions 4 and 10. Meanwhile, the coalition formation algorithm just requires *Neighbour Information Consistency Assumption (NICA)* (i.e., local information needs to be known by

Algorithm 4 Integrated decision-making framework

$k_0 = 0$;
while $k_0 \leq T - T_H$ **do**
  // Coalition Formation
    for all $a_i \in \mathcal{A}$ (in parallel) **do**
      Compute $\bar{d}_{ij}$ and $\bar{c}_{ij} = f_c^i(\bar{d}_{ij})$ for $\forall t_j \in \mathcal{T}$
    end for
    $\{\mathcal{S}_j\}$:= the partition resulted by Algorithm 1
  // Position Allocation
    for all $a_i \in \mathcal{S}_j$, $\forall t_j \in \mathcal{T}$ (in parallel) **do**
      Compute $\tilde{w}_{im}$ using (6.36) for $\forall p_m \in \mathcal{P}_j$
      $\{y_{im}\}$:= solution to Problem 3
      $p_i^* = p_m$ such that $y_{im} = 1$
    end for
  // Path planning
    $\mathbf{x}_i[k]$:= the trajectories by Algorithm 3, $\forall a_i \in \mathcal{A}$
    for $\forall k \in \{k_0, ..., k_0 + T_H\}$
  // Path following
    Follow $\mathbf{x}_i[k]$ for $\forall k \in \{k_0, ..., k_0 + T_H\}$, $\forall a_i \in \mathcal{A}$
    Update k_0 and $\mathbf{x}_{i,k_0}$ to current time, $\forall a_i \in \mathcal{A}$
end while

only one neighbour agent at each iteration) under a simple strongly-connected communication network [22], and inter-agent communication is even not necessary for the position allocation process. As long as local-information consistency required for the path planning process is guaranteed, the integrated framework can work asynchronously. Therefore, considering the fact that the path planning process is dominant in terms of environmental requirements, the experimental validation of the MPC-SCP [19] suggests the real-time implementability of the proposed framework. More rigorous analysis regarding detrimental effects of asynchronous agents will be subject to in our future study.

For now, let us discuss about the suboptimality of the proposed integrated frame-

work.

Proposition 6 (*Suboptimality of the integrated framework*)**.** *Let $r_{\mathcal{P}_j}$ denote the maximum radius from the central position of task t_j (i.e., $p_{j,0}$) to any working positions for the task (i.e., $\forall p_m \in \mathcal{P}_j$). For an instance of Problem 2 with setting $\bar{c}_{ij} := f_c(\bar{d}_{ij} - r_{\mathcal{P}_j})$, called* the dummy problem*, suppose that we have its LHR objective function value, denoted by $J^\star_{LHR}$. Then, the suboptimality of an outcome from the integrated framework (i.e., Algorithm 4) is lower bounded by $J_A/J^\star_{LHR}$, where J_A is the outcome's objective function value.*

Proof. Let J_{OPT} and $J^\star_{OPT}$ denote the optimal objective function value of Problem 1 and that of the dummy problem, respectively. We will firstly show that $J^\star_{OPT} \geq J_{OPT}$. Work capacity $\bar{w}_{ij}$ in the dummy problem is $\bar{w}_{ij} = w_{i,0} - f^i_c(\bar{d}_{ij} - r_{\mathcal{P}_j})$, which is always greater than or equals to $w_{im} = \eta_m(w_{i,0} - f^i_c(d_{im}))$ in the original problem for $\forall p_m \in \mathcal{P}_j$. This is because: (a) for every agent a_i and task t_j,

$$\bar{d}_{ij} - r_{\mathcal{P}_j} \leq d_{im}, \quad \forall p_m \in \mathcal{P}_j;$$

(b) the dummy problem has work efficiency ratio $\eta_m = 1$. Hence, given the optimal objective function values for both problems, it follows that $J^\star_{OPT} \geq J_{OPT}$.

For the dummy problem, its LHR objective function value $J^\star_{LHR}$ can be found by Equation (6.33). This value upper bounds $J^\star_{OPT}$ according to Proposition 1. From this, it turns out that

$$J^\star_{LHR} \geq J^\star_{OPT} \geq J_{OPT}.$$

Hence, the suboptimality of the outcome from the integrated framework (i.e., J_A/J_{OPT}) is lower bounded by $J_A/J^\star_{LHR}$. □

6.8 Application to Cooperative Jamming

6.8.1 Cooperative Stand-in Radar Jamming

We implement the proposed framework into a *cooperative stand-in jamming mission*. This mission can be categorised as "escort jamming", in which multiple aerial robots

with ECM (electro counter measure) transmitters are to neutralise given target radars to protect their allies being far behind. The strategy for stand-in jamming differs from that for typical *stand-off* jamming in that the aerial robots penetrate into enemy territory and jam the targets while being located nearby. This comparison can be illustrated as in Figure 6.4.

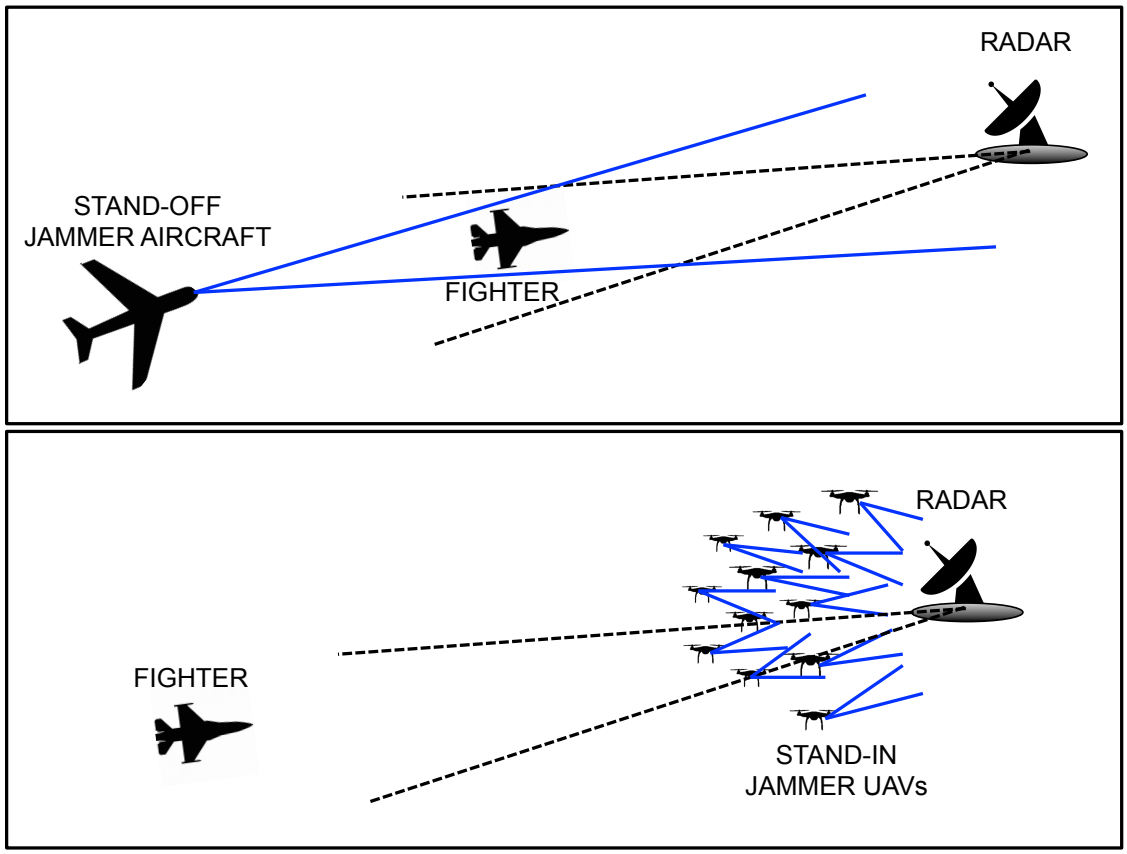


Figure 6.4: *Stand-off* Jamming vs. Cooperative *Stand-in* Jamming

A swarm of small-sized UAVs is suitable to carry out this mission because such UAVs can take advantage of proximity effect by being close enough to the adversary radars thanks to their small RCS (Radar Cross Section) i.e., lower observability, as well as fault tolerance as a multi-agent system. Despite their limited jamming resources, they can cooperatively provide enough jamming effectiveness by superimposing their signal strengths.

For now, we give a brief description of our prototype mission scenario. A set of micro UAVs are deployed to cooperatively jam a set of adversary radars. They begin a decision-making process for coalition formation and position allocation, and then transition to the assigned positions while avoiding any collision. After the agents succeed in jamming the radars, the allies, which are armed with Air-to-Ground Missiles (AGMs) and loitering until that moment, directly fly to and attack the radars. Note that the agents are assumed to have small RCSs due to their small sizes, thus not being tracked by the radars.

The cooperative jamming effectiveness by robotic coalition $\mathcal{S}_j$ towards the j-th radar is a function of the *Jamming to Signal ratio* as follows [31]:

$$(\mathsf{J}/\mathsf{S})_j = \frac{\sum_{\forall a_i \in \mathcal{S}_j} \mathsf{J}_{i,j}}{\mathsf{S}_j}. \tag{6.41}$$

Here, S_j is the backscattered signal strength to the j-th radar from one of the protected allies with the largest RCS:

$$\mathsf{S}_j = \frac{\mathsf{k}_j^R \cdot \sigma(\theta_{a,j})}{\mathsf{d}_{a,j}^4}, \tag{6.42}$$

where k_j^R is the radar-dependent coefficient, which varies with the radar's characteristics such as its transmitted power, wavelength, and antenna gains; $\mathsf{d}_{a,j}$ is the distance from the j-th radar to where the ally can be close to the radar while being protected; and σ is the RCS of the ally, which depends on its attitude with respect to the j-th radar (denoted by $\theta_{a,j}$).

The individual jamming signal strength of the i-th agent towards the j-th radar is defined by

$$\mathsf{J}_{i,j} = \frac{\mathsf{k}_i^A \cdot \mathsf{G}_{i,j}^R}{\mathsf{d}_{i,j}^2}, \tag{6.43}$$

where k_i^A is the agent-dependent coefficient; $\mathsf{d}_{i,j}$ is the distance between the i-th agent and the j-th radar when the agent performs jamming; and $\mathsf{G}_{i,j}^R$ is the radar antenna gain, which relies on the agent's position with respect to the radar's main-lobe.

The cooperative jamming is successful if $(\mathsf{J}/\mathsf{S})_j$ exceeds the radar's *burn-through value* $(\mathsf{J}/\mathsf{S})_j^{burn}$, which is determined by the radar's characteristics and signal processing method.

6.8.2 Implementation and Settings

In this numerical simulation, we have $n_a = 50$ of UAVs, $n_t = 3$ of tasks (i.e., target radars). For the radar and jamming-agent relevant coefficients prescribed in the previous section, the realistic values shown Table 6.3 are used. Note that k^A of micro-sized UAV is assumed from consideration of other UAV types' values. We uniform randomly generate $\mathsf{k}_i^A \in [0.025, 0.050]$ for the agents, and set that $\{\mathsf{k}_1^R, \mathsf{k}_2^R, \mathsf{k}_3^R\} = \{2 \cdot 10^9, 1.5 \cdot 10^9, 1 \cdot 10^9\}$ for the radars.

Table 6.3: Parameters of Jamming UAVs and Adversary Radars [31]

UAV size	k^A	Radar type	k^R
micro	0.05	short-range	$2 \cdot 10^7$
small	0.25	med.-range	$2 \cdot 10^8$
large	1	long-range	$2 \cdot 10^9$
$(\mathsf{J}/\mathsf{S})^{burn}$	G^R main-lobe	G^R side-lobe	
1	1	0.001	
1	1	0.001	
1	1	0.001	

We set that the agents can approach to the tasks as close to 60 m as possible, denoted by $\mathsf{d}_{\min}$, and they may not be located in the radars' main lobes: from (6.43), each agent a_i's initially-available work resource is set by

$$w_{i,0} = \frac{\mathsf{k}_i^A \cdot \mathsf{G}_{\text{side}}^R}{\mathsf{d}_{\min}^2}, \tag{6.44}$$

where the corresponding radar antenna gain, denoted by $\mathsf{G}_{\text{side}}^R$, is set to 0.001. Then, work efficiency ratio η_m for position $p_m \in \mathcal{P}_j$ is set by

$$\eta_m = \frac{\mathsf{d}_{\min}^2}{(\mathsf{d}_{\min} + \Delta\mathsf{d}_m)^2}, \tag{6.45}$$

where $\Delta \mathsf{d}_m$ is the distance from the working position p_m to the circle surrounding the j-th radar with the radius of $\mathsf{d}_{\min}$, as illustrated in Figure 6.5.

We set that as an agent uses its resource to transition, its given k^A is reduced: the decreasing rate is 0.005 per km. From this, we can define each agent's cost-to-go function: $f_c^i(d) = C_1 \cdot d$, where $C_1 = \frac{\mathsf{G}^R_{\text{side}}}{\mathsf{d}^2_{\min}} \cdot 0.005$.

We assume that the allies to protect are F-16 fighter, whose RCS is known as 5 m^2, and they should be protected until reaching 20 km before the radars (i.e., $\mathsf{d}_{a,j}$). From Equations (6.41) and (6.42), each task t_j's minimum requirement is set by

$$R_j = \frac{\mathsf{k}^R_j \cdot \sigma(\theta_{a,j})}{\mathsf{d}^4_{a,j}} \cdot (\mathsf{J/S})^{burn}_j. \tag{6.46}$$

In addition, we have $n_o = 2$ of obstacles. The inter-agent collision-avoidance radius is $r_{col} = 15\ m$. For every obstacle, its radius is $r_{obs,q} = 60\ m$. For each agent a_i, we consider 2-dimensional space and the point-mass kinematics model:

$$\dot{\mathbf{x}}_i(\mathsf{t}) = A\mathbf{x}_i(\mathsf{t}) + B\mathbf{u}_i(\mathsf{t}), \tag{6.47}$$

where

$$A = [\mathbf{0}_{2\times 2}\ I_{2\times 2}; \mathbf{0}_{2\times 2}\ \mathbf{0}_{2\times 2}],$$

$$B = [\mathbf{0}_{2\times 2}\ I_{2\times 2}]^\top,$$

$$\mathbf{u}_i(\mathsf{t}) = [u_1(\mathsf{t})\ u_2(\mathsf{t})]^\top.$$

The maximum speed and acceleration of every agent is $V_{\max} = 10\ m/s$ and $U_{\max} = 3\ m/s^2$, respectively.

We set that the agents generate collision-free trajectories over the time-horizon $T_H = 30$ seconds, but update them at every 20 seconds. For the time-horizon, the discretised time gap is set to $\Delta \mathsf{t} = 1$ sec. In order to reduce computation time, we use $\Delta \mathsf{t} = 10$ sec for the time range outside T_H. It is assumed that before $\mathsf{t} = 0$ the agents already finished Algorithm 4 for the first time-horizon, in which they compute for the next time-horizon, and so forth. The mission final time is $\mathsf{t}_f = 180$ sec. We set that $\epsilon_{CF} = 0.05$, $\epsilon_{SCP} = 2.5$ and $\lambda = 0.5$.

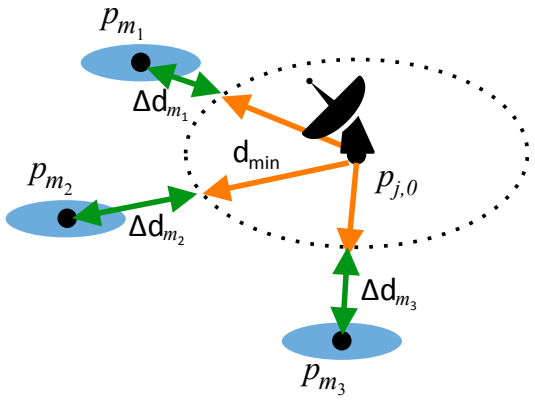


Figure 6.5: Definitions of $\mathsf{d}_{\min}$ and $\Delta \mathsf{d}_m$

In order to see how adaptively the agents using the proposed framework change their behaviours, we consider a dynamic environment in which some of the agents are lost in the middle of the mission: a half of agents assigned to task t_1 are randomly failed at $\mathsf{t} = 35$ sec; even so, the communication network of the remaining alive agents still holds Assumption 4; and every agent can notice the failure using the network shortly.

6.8.3 Results

Figure 6.6 show sequential snapshots of the resulted behaviours of the agents using the proposed framework. Each subfigure shows the locations of the agents, the tasks, the working positions and the obstacles at a certain time of the mission. The tasks are represented as blue, red, and green circles in the right, and the obstacles as black circles in the middle. The grey small circles around each task are the corresponding working positions. The size of the solid circle for each task indicates its minimum workload requirement, and that of the dashed black circle represents the boundary to which the agents can be as close as possible, i.e., $\mathsf{d}_{\min}$. Each agent and its collision-avoidance radius are illustrated as a coloured dot and the circle surrounding it, respectively. Here, the circle is coloured by the colour of the agent's assigned task. The size of each dot indicates the agent's currently-available work resource, and it shrinks as the resource is consumed by transitioning towards its assigned working position. The dotted trail

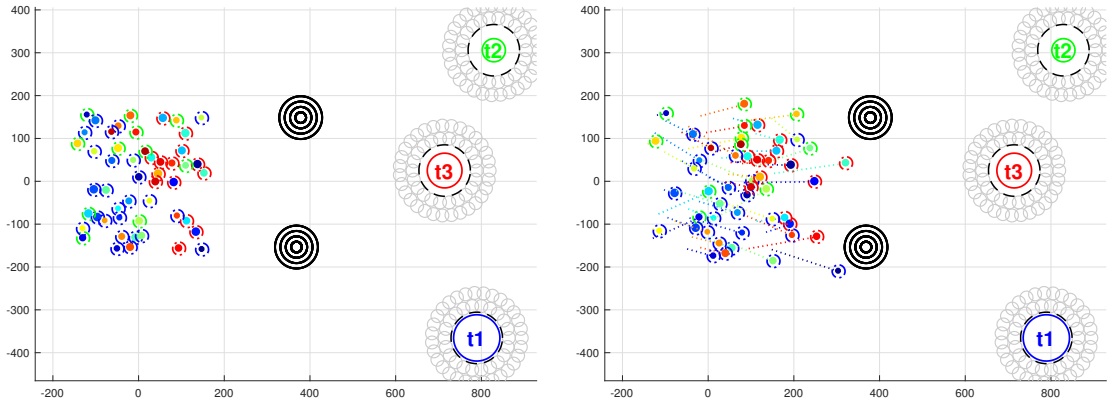
behind each agent is the agent's previous trajectory.

As presented in Figure 6.6(a)–(b), the agents follow the trajectories towards their assigned positions from $\mathtt{t} = 0$ to $\mathtt{t} = 20$. At $\mathtt{t} = 35$, the half of the agent assigned to task t_1 are somehow failed (Figure 6.6(c)). Note that the lost agents and their trajectories are represented as black-dotted circles and lines, respectively. Due to the update interval for new trajectories, by $\mathtt{t} = 60$, the remaining alive agents finish to reassign the positions and generate new trajectories for the next time horizon (Figure 6.6(d)). Here, some of agents previously assigned to tasks t_2 and t_3 are assigned to t_1. After that, the alive agents follow the new collision-free trajectories until the mission final time, as presented in Figure 6.6(e)–(f). Figure 6.6(f) shows that the agents are sensibly allocated with consideration of their working resources and the tasks' requirements.

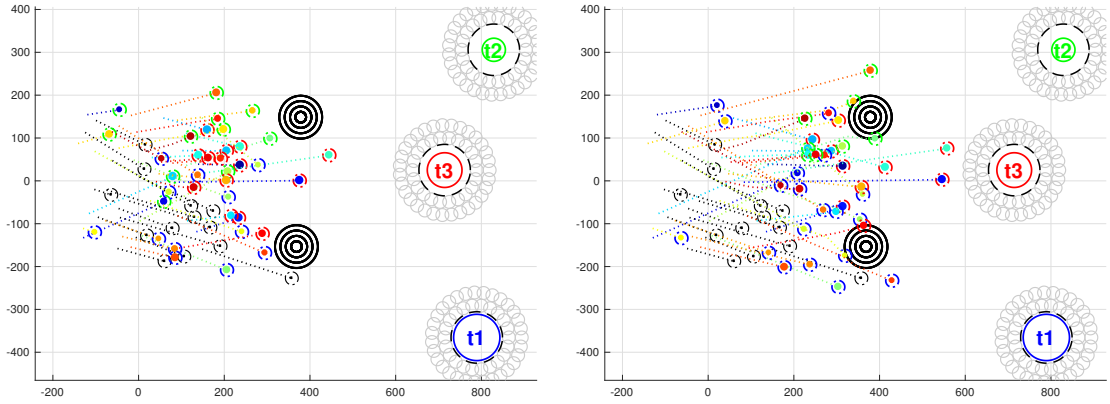
Figure 6.7 shows each agent's minimum distance from its closest neighbour during the entire mission. Each line is coloured corresponding to the agent colour of Figure 6.6. This result indicates that all the agents using the proposed framework comply with the inter-agent collision avoidance constraints.

Table 6.4 presents the computation time spent to obtain the decision-making result. Because the framework sequentially solves an optimisation problem over the time horizon, the computation time shown in the table is the average over every optimisation. Note that the computation times spent for the coalition formation and position allocation are only significant when the agents planned over the time periods $\mathtt{t} \in \{[0, 20], [60, 80]\}$, thus we only used the corresponding computation times to obtain the average value. All the simulations were performed using MATLAB R2016b on a computer (Mac mini Late 2014) with Intel Core i5 2.8 GHz, 16GB Memory, and OS X Yosemite v.10.10.5. To solve convex optimisations (i.e., Problem 6) in Algorithm 3, CVX ver. 2.1 [32] was utilised. The table shows that approximately 6 seconds are required for each time horizon. Since the agents in the experiment updated their trajectories at every 20 seconds, there still remain approximately 14 seconds extra.

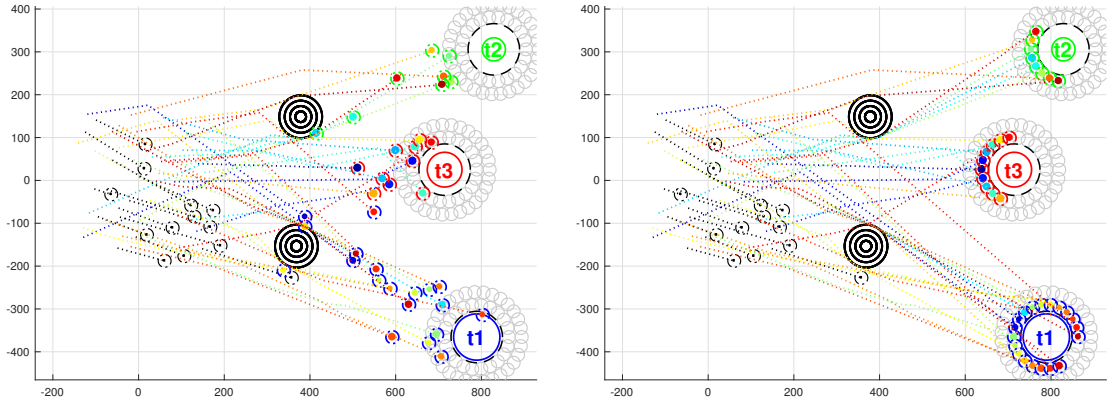
In addition, Table 6.4 shows the suboptimality of the outcome from the proposed framework. We exploited Proposition 6 to obtain the suboptimality lower bound, which indicates that the outcome is near-optimal.



(a) t = 0 **(b)** t = 20



(c) t = 35 **(d)** t = 60



(e) t = 120 **(f)** t = 180

Figure 6.6: The resulted behaviours of the agents using the proposed framework ($n_a = 50$, $n_t = 3$, and $n_o = 2$).

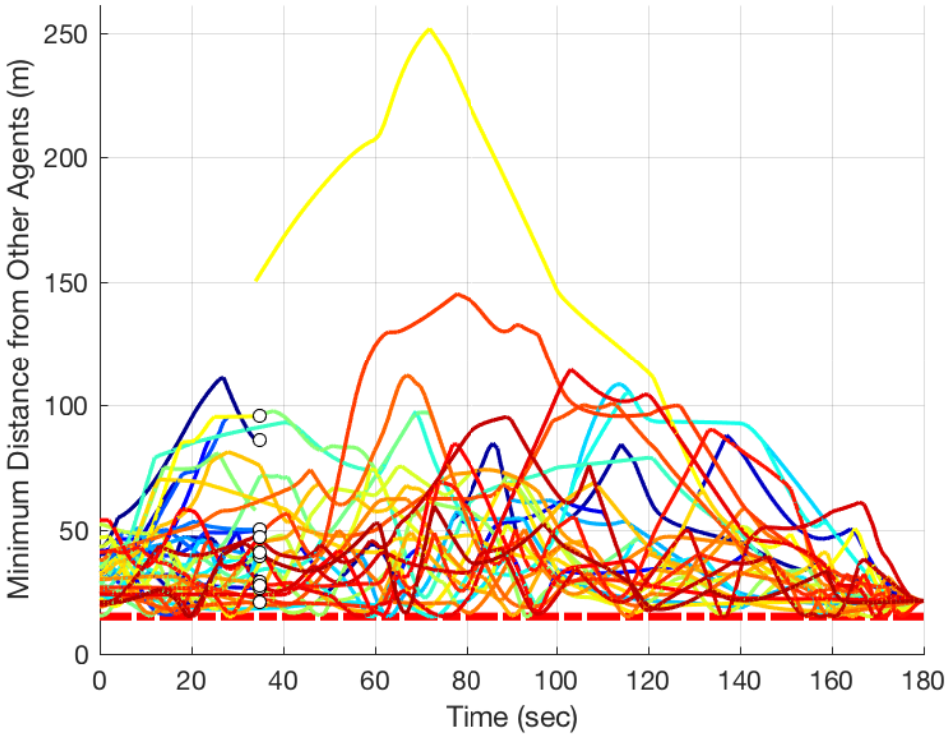


Figure 6.7: Each agent's minimum distance from its closest neighbour agent during the mission time

6.9 Conclusion

This study addressed a swarm of heterogeneous robots' decision-making issues including team formation, team-to-task assignment, agent-to-working-position selection, fair resource allocation considering tasks' minimum requirements, and trajectory optimisation with collision avoidance. The proposed framework decouples the complex original problem into three subproblems (i.e., coalition formation, position allocation, and path planning) and deals with them sequentially by three different subroutine algorithms in a decentralised manner. For the coalition formation subproblem, we introduced the game-theoretical method that recursively sets the minimum RSI and minimises given agents' unnecessary costs (equivalently maximises their work capacities). We showed

Table 6.4: Simulation results ($n_a = 50$, $n_t = 3$, $n_o = 2$)

Subroutine	Computation time (sec)
Coalition formation (Algorithm 1)	0.101
Position allocation	0.045
Path planning (Algorithm 3)	5.088
Suboptimality of Algorithm 4	**(%)**
Lower bound	97.42

that the position allocation subproblem can be solved by a simple sorting algorithm under reasonable assumptions. For the trajectory optimisation, we utilised the MPC-SCP algorithm by which the agents can generate collision-free trajectories over a time horizon. By introducing the LHR solution concept, we proposed a methodology to analyse suboptimality of the proposed framework. As a proof of concept, we implemented the proposed integrated framework into a UAV swarm's cooperative stand-in jamming mission scenario. It was suggested from the numerical experiment results that the framework could be computationally feasible, fault-tolerant, and near-optimal.

A natural progression of this study is to validate this framework in a real-robot experiment. Furthermore, a formal analysis regarding the algorithmic complexity must be a significant contribution.

Appendix

A. Linearisation and Discretisation

In order to address Problem 4, we firstly linearise the dynamics in (6.9) about the nominal trajectory $\bar{\mathbf{x}}_i$, which is assumed to be given.

$$\dot{\mathbf{x}}_i = A(\bar{\mathbf{x}}_i)\mathbf{x}_i + B\mathbf{u}_i + z(\bar{\mathbf{x}}_i) \tag{6.48}$$

where $A(\bar{\mathbf{x}}_i) = \frac{\delta \mathbf{f}}{\delta \mathbf{x}_i} \mid_{\bar{\mathbf{x}}_i}$ and $z(\bar{\mathbf{x}}_i) = \mathbf{f}(\bar{\mathbf{x}}_i) - \frac{\delta \mathbf{f}}{\delta \mathbf{x}_i} \mid_{\bar{\mathbf{x}}_i} \bar{\mathbf{x}}_i$. Then, we transform the problem to the discrete-time version. Regarding Equation (6.29), it follows from $||G\dot{\mathbf{x}}_i(\mathsf{t})||\ d\mathsf{t} = ||d\mathbf{p}_i(\mathsf{t})/d\mathsf{t}||\ d\mathsf{t}$ that

$$\begin{aligned} \int_{\mathsf{t}_0}^{\mathsf{t}_f} ||G\dot{\mathbf{x}}_i(\mathsf{t})||\ d\mathsf{t} &= \int_{\mathsf{t}_0}^{\mathsf{t}_f} ||d\mathbf{p}_i(\mathsf{t})/d\mathsf{t}||\ d\mathsf{t} \\ &\approx \int_{\mathsf{t}_0}^{\mathsf{t}_f - \Delta\mathsf{t}} ||\frac{\mathbf{p}_i(\mathsf{t}+\Delta\mathsf{t}) - \mathbf{p}_i(\mathsf{t})}{\Delta\mathsf{t}}||\ d\mathsf{t}, \end{aligned} \tag{6.49}$$

where $\Delta\mathsf{t}$ is the time difference for the discretisation. We set that, for $k = 0, 1, ..., T$,

$$\mathbf{x}_i[k] := \mathbf{x}_i(\mathsf{t}_k), \quad \mathbf{p}_i[k] := \mathbf{p}_i(\mathsf{t}_k), \quad \mathbf{u}_i[k] := \mathbf{u}_i(\mathsf{t}_k), \tag{6.50}$$

where $T := (\mathsf{t}_f - \mathsf{t}_0)/\Delta\mathsf{t}$ is the number of discrete time steps; $\mathsf{t}_T := \mathsf{t}_f$; $\mathsf{t}_{k_0} := \mathsf{t}_0$; and $\mathsf{t}_{k+1} := \mathsf{t}_k + \Delta\mathsf{t}$ for all k. Then, Equation (6.49) becomes $\sum_{k=k_0}^{T-1} ||\mathbf{p}_j[k+1] - \mathbf{p}_j[k]||$. Finally, the right term in Equation (6.29) becomes

$$\min_{\mathbf{u}_i} \sum_{k=k_0}^{T-1} ||G(\mathbf{x}_i[k+1] - \mathbf{x}_i[k])||. \tag{6.51}$$

Likewise, Equation (6.48) can be reduced to

$$\mathbf{x}_i[k+1] = A_i[k]\mathbf{x}_i[k] + B_i[k]\mathbf{u}_i[k] + z_i[k], \quad k = k_0, ..., T-1, \tag{6.52}$$

where

$$A_i[k] = e^{A(\bar{\mathbf{x}}_i(\mathsf{t}_k))\Delta\mathsf{t}},\ B_i[k] = \int_0^{\Delta\mathsf{t}} e^{A(\bar{\mathbf{x}}_i(\mathsf{t}_k))\tau} B d\tau,$$

$$z_i[k] = \int_0^{\Delta\mathsf{t}} e^{A(\bar{\mathbf{x}}_i(\mathsf{t}_k))\tau} z(\bar{\mathbf{x}}_i(\mathsf{t}_k)) d\tau.$$

Equations (6.10), (6.30), (6.12)–(6.15) can be also written in discretised form as follows:

$$\mathbf{x}_i[k_0] = \mathbf{x}_{i,0}, \tag{6.53}$$

$$\mathbf{x}_i[T] = [p^*; \mathbf{0}_{n_d \times 1}], \tag{6.54}$$

$$||G(\mathbf{x}_i[k] - \mathbf{x}_l[k])|| \geq r_{col}, \quad k = k_0, ..., T, \ \forall a_l \in \mathcal{A} \setminus \{a_i\}, \tag{6.55}$$

$$||G\mathbf{x}_i[k] - \mathbf{o}_q|| \geq r_{obs,q}, \quad k = k_0, ..., T, \ \forall o_q \in \mathcal{O}, \tag{6.56}$$

$$||\mathbf{u}_i[k]|| \leq U_{max}, \quad k = k_0, ..., T, \tag{6.57}$$

$$||H\mathbf{x}_i[k]|| \leq V_{max}, \quad k = k_0, ..., T, \tag{6.58}$$

B. Minimisation Knapsack Problem (MinKP)

Our proposed approach use an algorithm for *the 0/1 minimisation knapsack problem* (MinKP, for short) [24] as its subroutine. MinKP is defined as:

Definition 2 (*the 0/1 minimisation knapsack problem*)**.** Suppose that there are a knapsack with its minimum requirement R and a set of n items $Z = \{z_1, ..., z_n\}$, where each item z_i has its value v_i and cost c_i. The objective is to pack the knapsack with the items so that the total value of all inserted items exceeds the minimum requirement while minimising the resultant total cost:

$$\min_{\{x_i \in \{0,1\}\}} \quad \sum_{i=1}^{n} c_i x_i \qquad \text{s.t.} \quad \sum_{i=1}^{n} v_i x_i \geq R$$

where $x_i = 1$ if item z_i is inserted in the knapsack. Let $\textsc{MinKP}(Z, R, \mathcal{V}, \mathcal{C})$ denote an algorithm for MinKP, where $\mathcal{V} = \{v_i\}$ and $\mathcal{C} = \{c_i\}$ are sets of the items' values and costs, respectively. The output of this algorithm is the set of selected item for the knapsack.

C. Distributed Mutex Subroutine

For Algorithm 2, we use the distributed mutex algorithm proposed in our previous work [26], the detail of which is described in Algorithm 5. The algorithm makes sure

Algorithm 5 Distributed Mutex Subroutine [26]

function D-MUTEX($\mathcal{M}^i_{rcv}$)
  for each message $\mathsf{msg}^k \in \mathcal{M}^i_{rcv}$ **do**
    if $(r^i < r^k)$ or $(r^i = r^k \ \& \ s^i < s^k)$ **then**
      $r^i \leftarrow r^k$
      $s^i \leftarrow s^k$
      $\Pi^i \leftarrow \Pi^k$
      $\mathsf{satisfied}^i \leftarrow \mathsf{satisfied}^k$
    end if
  end for
  return $\{r^i, s^i, \Pi^i, \mathsf{satisfied}^i\}$
end function

that there is only one (local) partition that dominates (or will finally dominate depending on the communication network) any other partitions. In other words, multiple partitions locally evolve and some of them only eventually can survive at every main loop of Algorithm 2 even under asynchronous behaviours of agents as long as their communication network is at least strongly-connected. Even if we may encounter multiple Nash stable partitions at last, one of them can be distributedly selected by the agents.

References

[1] E. Sahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 10–20.

[2] I. Navarro and F. Matía, "An Introduction to Swarm Robotics," *ISRN Robotics*, vol. 2013, pp. 1–10, 2013.

[3] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm Robotics: a Review from the Swarm Engineering Perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[4] K. Barton and D. Kingston, "Systematic Surveillance for UAVs: A Feedforward Iterative Learning Control Approach," in *American Control Conference*, Washington, DC, USA, 2013, pp. 5917–5922.

[5] I. Bekmezci, O. K. Sahingoz, and S. Temel, "Flying Ad-Hoc Networks (FANETs): A Survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[6] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz, "Help from the Sky: Leveraging UAVs for Disaster Management," *IEEE Pervasive Computing*, vol. 16, no. 1, pp. 24–32, 2017.

[7] I. Jang, J. Jeong, H.-S. Shin, S. Kim, A. Tsourdos, and J. Suk, "Cooperative Control for a Flight Array of UAVs and an Application in Radar Jamming," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8011–8018, 2017.

[8] B. P. Gerkey and M. J. Matarić, "A Formal Analysis and Taxonomy of Task Allocation in Multi-robot Systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[9] G. A. Korsah, A. Stentz, and M. B. Dias, "A Comprehensive Taxonomy for Multi-robot Task Allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[10] H.-S. Shin and P. Segui-Gasco, "UAV Swarms: Decision-Making Paradigms," in *Encyclopedia of Aerospace Engineering*. John Wiley & Sons, 2014, pp. 1–13.

[11] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica*, vol. 33, no. 03, pp. 463–497, 2015.

[12] B. Saicharan, R. Tiwari, and N. Roberts, "Multi Objective optimization based Path Planning in robotics using nature inspired algorithms: A survey," in *IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems*, Delhi, India, 2016, pp. 1–6.

[13] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable Self-assembly in a Thousand-robot Swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[14] O. Shehory and S. Kraus, "Methods for Task Allocation via Agent Coalition Formation," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 165–200, 1998.

[15] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Image and animation display with multiple mobile robots," *International Journal of Robotics Research*, vol. 31, no. 6, pp. 753–773, 2012.

[16] M. Turpin, N. Michael, and V. Kumar, "CAPT: Concurrent assignment and planning of trajectories for multiple robots," *International Journal of Robotics Research*, vol. 33, no. 1, pp. 98–112, 2014.

[17] M. Turpin, K. Mohta, N. Michael, and V. Kumar, "Goal assignment and trajectory planning for large teams of interchangeable robots," *Autonomous Robots*, vol. 37, no. 4, pp. 401–415, 2014.

[18] D. Morgan, G. P. Subramanian, S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic Guidance of Distributed Systems using Sequential Convex Programming," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA, 2014, pp. 3850–3857.

[19] D. Morgan, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, "Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1261–1285, 2016.

[20] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, "Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 6, pp. 1725–1740, 2014.

[21] J. D.F. Votaw and A. Orden, "The Personnel Assignment Problem," in *Symposium on Linear Inequalities and Programming*, Washington, DC: Planning Research Division, Comptroller, Headquarters US Air Force, 1952, pp. 155–163.

[22] I. Jang, H.-S. Shin, and A. Tsourdos, "A Game-theoretical Approach to Heterogeneous Multi-robot Task Assignment Problem with Minimum Workload Requirements," in *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, Linkoping, Sweden, 2017, pp. 156–161.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithm*, 3rd ed. Cambridge, Massachusetts, USA: The MIT Press, 2009.

[24] M. M. Güntzer and D. Jungnickel, "Approximate Minimization Algorithms for the 0/1 Knapsack and Subset-Sum Problem," *Operations Research Letters*, vol. 26, no. 2, pp. 55–66, 2000.

[25] D. Beyer and R. Ogier, "Tabu Learning: A Neural Network Search Method for Solving Nonconvex Optimization Problems," in *IEEE International Joint Conference on Neural Networks*, 1991, pp. 953–961.

[26] I. Jang, H.-S. Shin, and A. Tsourdos, "Anonymous Hedonic Game for Task Allocation in a Large-scale Multiple Agent System," *IEEE Transactions on Robotics (conditionally accepted)*, 2018.

[27] H. W. Khun, "The Hungarian Method for the Assignment Problem," *Naval Reserach Logistic Quarterly*, vol. 2, pp. 83–97, 1955.

[28] K. Mehlhorn and P. Sanders, *Algorithms and Data Structures*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[29] Y. Qin, M. Cao, and B. D. O. Anderson, "Asynchronous Agreement through Distributed Coordination Algorithms Associated with Periodic Matrices," in *Proceedings of the IFAC World Congress*, Toulouse, France, 2017, pp. 1778–1783.

[30] L. B. Johnson, H.-L. Choi, and J. P. How, "The Role of Information Assumptions in Decentralized Task Allocation: A Tutorial," *IEEE Control Systems*, vol. 36, no. 4, pp. 45–58, 2016.

[31] J. Kim and P. Hespanha, "Cooperative radar jamming for groups of unmanned air vehicles," in *IEEE Conference on Decision and Control*, Atlantis, Bahamas, 2004, pp. 632–637.

[32] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, ver 2.1," 2017.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis has aimed to develop innovative and transformative decision-making frameworks that enable a robotic swarm to autonomously partition themselves into multiple disjoint teams to collaborate a set of relatively fewer but complicated tasks. For such a large-scale multi-agent system to be coordinated effectively, the frameworks should be executable based on local information in a decentralised manner, operable for a wide range of the system size, predictable in terms of collective behaviours, flexible to dynamic environments, operable asynchronously, and preferably able to accommodate heterogeneous agents [1–5].

Firstly, for a swarm of homogeneous robots, the thesis proposed two novel frameworks based on biological inspiration in Chapter 2 and game theories in Chapter 3, respectively. The former, called LICA-MC, is based on a Markov process in which population fractions of a swarm are modelled as the system state, and each agent behaves stochastically according to a time-inhomogeneous Markov matrix (i.e., stochastic policies) depending on the difference from the current swarm distribution to a desired status. A swarm of fish in nature inspired this work: despite insufficient awareness of the entire group, they can be well-coordinated (e.g., schooling, shoaling, milling) by sensing and maintaining social distance from neighbours in the group [6–9]. Analogously, each agent in the proposed framework only relies on local information and requires its local

consistency over neighbouring agents (i.e. *Local Information Consistency Assumption*) to adaptively generate the current stochastic policy: this is one of the main novelties compared with existing frameworks, where global information needs to be known by the entire agents (i.e., *Global Information Consistency Assumption*). Thanks to LICA, the proposed framework has various advantages, e.g., less inter-agent communication, a shorter timescale for using new information, and the potential to incorporate an asynchronous decision-making process. We proved that, even using such limited information, the agents can converge to the desired collective status while maintaining scalability, flexibility, and long-term system efficiency, comparable to a recently-proposed existing global-information-based approach [3]. We numerically showed that the proposed framework is robust in a realistic environment where information sharing over agents is partially and temporarily disconnected. Moreover, the thesis explicitly presented the design requirements to have all these advantages and provided specific implementation examples concerning travelling costs minimisation, over-congestion avoidance, and quorum models, respectively.

The latter, called GRAPE, is based on anonymous hedonic games, where each robot is regarded as a selfish agent attempting to join the most preferred coalition amongst all in accordance with its individual preferences regarding the size of each coalition. The thesis showed that selfish agents who have social inhibition (more specifically, whose individual interests are transformable to SPAO preferences) can always converge to a Nash stable partition by using the proposed decentralised decision-making framework. The framework is straightforward and executable based on local interactions with neighbour agents under a strongly-connected communication network and even in asynchronous environments because it only relies on NICA (*Neighbour Information Consistency Assumption*), i.e., an agent's local information only needs to be known by at least one of its neighbouring agents. The framework has a recursive process with polynomial-time complexity $O(n_a^2 d_G)$, where n_a is the number of agents and d_G is the graph diameter of the agents' communication network. Even when it comes to a centralised version (i.e., the case when $d_G = 1$), the analysed complexity is still lower than that of the existing centralised algorithm in [10] (i.e., $O(n_a^2 n_t)$, where n_t is the number of tasks). Furthermore, it was experimentally shown that the proposed framework in practice could run rapidly with having its complexity of almost $O(n_a)$. The thesis an-

alytically presented a mathematical formulation for computing a converged outcome's minimum-guaranteed suboptimality, and numerically showed that the lower bound was averagely 60% to 70%, depending on instances. We additionally showed that 50 % of suboptimality is guaranteed if social utilities are non-decreasing functions with respect to the number of coworking agents. Our numerical experiments confirmed that the proposed framework is scalable, fast adaptable to dynamical environments, and robust even in a realistic situation where some of the agents temporarily halt operation during a mission.

Then, in Chapter 4, the thesis compared the characteristics of the bio-inspired framework and the game-theoretical framework. For the comparison, we implemented both frameworks for a problem of robotic swarm labour division, introduced evaluation metrics concerning their convergence performances, then performed numerical experiments with various environmental settings. The statistical results showed that LICA-MC provides excellent scalability with respect to the number of robots, whereas GRAPE has polynomial complexity but is more efficient in terms of convergence time (particularly when accommodating a relatively fewer number of robots) as well as total travelling costs. This is because GRAPE makes a global agreed plan before action, whereas agents in LICA-MC myopically generate and follow their local policies without confirming any social agreement. Additional sensitivity analysis suggested that possible traffic congestion may degrade the performance of GRAPE, which is not the case for LICA-MC owing to its built-in path planning function, meanwhile LICA-MC is vulnerable to the slower mobility of each robot. On the other hand, GRAPE does not need the specific description of a desired labour distribution status (i.e., as long as agents are given information about tasks, they can find a social agreement based on their individual preferences), which is contrarily required to be pre-known to agents of LICA-MC. More importantly, absolute preferences over tasks are only able to be incorporated in LICA-MC, whereas agents in GRAPE can have different individual preferences: this feature provides GRAPE the potential to accommodate heterogeneous agents to some extent.

In Chapter 5, the thesis attempted to extend GRAPE to incorporate heterogeneous agents because of the potential found in Chapter 4. The heterogeneity of agents possibly occur in practice even for a swarm consisting of homogeneously-manufactured

robots, for example, because of different levels of battery remained. We considered the case where each task has its minimum workload requirement to be fulfilled by multiple agents, and the agents have different work capacities and costs depending on the tasks. The decision-making objective is to find an assignment that minimises the total cost of assigned agents while satisfying the requirements, which was formulated as MinGAP-MR. This optimisation problem reflects a business concern regarding reducing unnecessary costs but marginally complying with customers' requirements. Since it was not possible to directly use GRAPE for the problem due to the heterogeneity, we adopted tabu-learning heuristics where an agent penalises its previously chosen coalition whenever it changes a decision: this variant is called T-GRAPE. We proved that, by doing so, a Nash stable partition is always guaranteed to be determined in a decentralised manner. Furthermore, our experimental results revealed that an outcome's suboptimality lower bound is at least greater than 50%, and the number of required iterations until convergence remains the same order of the number of given agents.

Finally, Chapter 6 addressed additional practical issues in decision-making when a robotic swarm is utilised for a cooperative mission: heterogeneous robots' team formation, team-to-task assignment, agent-to-working-position selection, fair resource allocation considering tasks' minimum requirements, and trajectory optimisation with collision avoidance. We proposed an integrated framework that decouples the complex original problem into three subproblems (i.e., coalition formation, position allocation, and path planning) and deals with them sequentially by three different subroutine algorithms. The coalition formation module based on T-GRAPE deals with a max-min problem, the objective of which is to partition the agents into disjoint task-specific teams in a way that balances the agents' work resources in proportion to the task's requirements. For agents assigned to the same task, given reasonable assumptions, the position allocation subproblem can be efficiently addressed in terms of computational complexity. For the trajectory optimisation, we utilised the MPC-SCP algorithm which reduces the size of the problem so that the agents can generate collision-free trajectories on a real-time basis. The integrated framework works under a non-fully-connected network of the agents and by their asynchronous behaviours. As a proof of concept, we implemented the proposed integrated framework into a cooperative stand-in jamming mission scenario. It was suggested from the numerical experiment results that

Table 7.1: Comparison of the proposed task allocation frameworks

	LICA-MC	GRAPE	T-GRAPE
Decentralisation	◯	◯	◯
Scalability	$O(1)$	$O(n_a^2)$	?[1]
Predicability	◯	◯	?
Adaptability	◯	◯	◯
Asynchronisation	△[2]	◯	◯
(Information sharing level required	LICA	NICA	NICA)
Heterogeneity	×	△[3]	◯

[1]Similar to GRAPE in empirical experiments, but not confirmed analytically.
[2]On the basis of agent subgroups
[3]Agents with different individual interests should be concerned with the cardinality of other agents

the framework could be computationally feasible, fault-tolerant, and near-optimal.

In summary, this research has developed multiple decision-making frameworks for a robotic swarm's task allocation problem. We proposed LICA-MC and GRAPE for a homogeneous robotic swarm, and then, based on the latter, we addressed heterogeneous agents by T-GRAPE. All the task allocation frameworks can be operated in a decentralised manner based on local information, but technically there exist differences. LICA-MC relaxes global information consistency assumption, but still requires agents to collect local information from all the others in their neighbour tasks for each time step (literally, LICA). Thus, its asynchronous operation is only possible on the basis of agent subgroups (i.e., an agent needs to wait for local information to be shared within its coalition at least). Meanwhile, in GRAPE or T-GRAPE, each agent's decision-making process can be independently performed even when local information from a single neighbour agent is only available (i.e., NICA). Therefore, the frameworks are executable in a fully-asynchronous manner. Table 7.1 summarises how the proposed frameworks comply with the desired features presented in Chapter 1.

In various domains apart from robotics, generic multi-agent systems have also been considered as promising solutions, and thus we believe that the proposed frameworks can be exploited to address some of their decision-making problems directly or with some modifications. In particular, we have high expectations on GRAPE because, as long as it is able to model SPAO preferences from given agents, one can benefit from all the advantages of GRAPE presented in the thesis. For example, in the coordination of self-driving cars, their individual interests regarding available lanes can be extracted to SPAO preferences with consideration of their myopic future intentions such as turning left, right, or keeping straight. We expect that this application of GRAPE could improve the traffic quality. For more generalised problems, the game-theoretical framework with tabu-learning heuristics (i.e., T-GRAPE) would be useful to find a social agreement of any heterogeneous agents. In fact, using the framework, we dealt with a frequency channel assignment problem of networked multiple UAVs in [11] (i.e., the paper C4 in Section 1.5). This problem is, as a resource allocation problem, such that the communication network of agents should be linked in a way that minimises communicational interferences given limited frequency band. For the case where scalability is essential and homogeneous agents are considered, LICA-MC would be much attractive. Particularly, if the time scale for state transition is relatively inconsiderable compared with those for computation and communication (e.g., in the domain of parallel computing), it is highly recommended to use LICA-MC. In robotics, continuous movement of agents in the framework may degrade the system performance in terms of fuel consumption. However, this would be more preferred in some cases, for military examples, where the agents have to avoid attacks or where they are required to deceive foes.

7.2 Future Work

Future work of this study, particularly regarding GRAPE, is to relax anonymity of agents and thus to consider a combination of the agents' identities. Empirical results showed that heterogeneous agents with social inhibition also could converge to a Nash stable partition if they share social utilities in a weighted balanced manner. Rigorous analysis regarding their converging behaviours and the corresponding outcome's suboptimality must be significant contributions. Another progression is to analyse the

quality of a Nash stable partition obtained by GRAPE in terms of fair task allocation because our various experiments showed that the outcome provides individual utilities to agents in a balanced manner. For T-GRAPE, more research is required to analytically evaluate its suboptimality, especially connecting with the approximation ratio of its subroutine for MinKP, and the upper bound of possible iterations. For LICA-MC, it would be interesting to relax the required local communication between neighbour agents by incorporating a vision-based local density estimation such as [12].

Since this research assumed that high-level mission descriptions from human operators are already given, future research will focus on developing a human-robot interface technique and linking this with decision-making frameworks for multiple robots. For example, human operators directly control a subset of the robots, called leaders, and under a swarm intelligence framework, the other remaining robots are to be influenced in a way that a desired collective behaviour is generated quicker than usual. Here, we need to consider how to select the swarm leaders and their effects to the other remaining robots in terms of convergence rate and the system performance. Besides, we may encounter teleoperational stability issues, which are known as difficult subjects even for a single robot. Possible delayed input signals from the human to the leaders may give rise to unexpected collective outcomes. Lastly, since a large number of robots are given, it is not straightforward to represent the entire robots' status as something understandable to the human operators. Development of a useful swarm representation will also be another future work.

References

[1] E. Sahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 10–20.

[2] M. Dorigo, M. Birattari, and M. Brambilla, "Swarm robotics," *Scholarpedia*, vol. 9, no. 1, p. 1463, 2014.

[3] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1103–1123, 2017.

[4] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm Robotics: a Review from the Swarm Engineering Perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[5] L. Johnson, S. Ponda, H.-L. Choi, and J. How, "Asynchronous Decentralized Task Allocation for Dynamic Environments," in *Infotech@Aerospace*, St.Louis, MO, USA, 2011.

[6] B. L. Partridge, "The Structure and Function of Fish Schools," *Scientific American*, vol. 246, no. 6, pp. 114–123, 1982.

[7] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective Leadership and Decision-making in Animal Groups on the Move," *Nature*, vol. 433, no. 7025, pp. 513–516, 2005.

[8] C. Becco, N. Vandewalle, J. Delcourt, and P. Poncin, "Experimental Evidences of a Structural and Dynamical Transition in Fish School," *Physica A: Statistical Mechanics and its Applications*, vol. 367, pp. 487–493, 2006.

[9] J. Gautrais, C. Jost, and G. Theraulaz, "Key Behavioural Factors in a Self-Organised Fish School Model," *Annales Zoologici Fennici*, vol. 45, no. 5, pp. 415–428, 2008.

[10] A. Darmann, "Group Activity Selection from Ordinal Preferences," in *Algorithmic Decision Theory. ADT 2015. Lecture Notes in Computer Science*, ser. Lecture

Notes in Computer Science, T. Walsh, Ed. Berlin, Heidelberg: Springer Cham, 2015, vol. 9346, pp. 35–51.

[11] H.-S. Shin, I. Jang, and A. Tsourdos, "Frequency Channel Assignment for Networked UAVs using a Hedonic Game," in *RED-UAS 2017*, Linkoping, Sweden, 2017.

[12] S. A. M. Saleh, S. A. Suandi, and H. Ibrahim, "Recent Survey on Crowd Density Estimation and Counting for Visual Surveillance," *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 103–114, 2015.

Bibliography

References for Chapter 1

[1] H.-S. Shin and P. Segui-Gasco, "UAV Swarms: Decision-Making Paradigms," in *Encyclopedia of Aerospace Engineering*. John Wiley & Sons, 2014, pp. 1–13.

[2] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable Self-assembly in a Thousand-robot Swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[3] E. Sahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 10–20.

[4] I. Navarro and F. Matía, "An Introduction to Swarm Robotics," *ISRN Robotics*, vol. 2013, pp. 1–10, 2013.

[5] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm Robotics: a Review from the Swarm Engineering Perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[6] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot Task Allocation: A Review of the State-of-the-Art," in *Cooperative Robots and Sensor Networks 2015*. Cham: Springer International Publishing, 2015, vol. 604, pp. 31–51.

[7] M. Dorigo, M. Birattari, and M. Brambilla, "Swarm robotics," *Scholarpedia*, vol. 9, no. 1, p. 1463, 2014.

[8] BBC News, "CES 2018: Intel's swarm of drones lights up Vegas night sky," 2018. [Online]. Available: http://www.bbc.co.uk/news/av/technology-42643790/ces-2018-intel-s-swarm-of-drones-lights-up-vegas-night-sky

[9] K. Barton and D. Kingston, "Systematic Surveillance for UAVs: A Feedforward Iterative Learning Control Approach," in *American Control Conference*, Washington, DC, USA, 2013, pp. 5917–5922.

[10] S. Hauert, J.-C. Zufferey, and D. Floreano, "Evolved swarming without positioning information: an application in aerial communication relay," *Autonomous Robots*, vol. 26, no. 1, pp. 21–32, 2009.

[11] I. Bekmezci, O. K. Sahingoz, and S. Temel, "Flying Ad-Hoc Networks (FANETs): A Survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[12] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz, "Help from the Sky: Leveraging UAVs for Disaster Management," *IEEE Pervasive Computing*, vol. 16, no. 1, pp. 24–32, 2017.

[13] Jianing Chen, M. Gauci, Wei Li, A. Kolling, and R. Gros, "Occlusion-Based Cooperative Transport with a Swarm of Miniature Mobile Robots," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 307–321, 2015.

[14] M. Mayer, "The new killer drones: understanding the strategic implications of next-generation unmanned combat aerial vehicles," *International Affairs*, vol. 91, no. 4, pp. 765–780, 2015.

[15] D. Albani, D. Nardi, and V. Trianni, "Field coverage and weed mapping by UAV swarms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vancouver, BC, Canada, 2017, pp. 4319–4325.

[16] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1103–1123, 2017.

[17] L. Johnson, S. Ponda, H.-L. Choi, and J. How, "Asynchronous Decentralized Task Allocation for Dynamic Environments," in *Infotech@Aerospace 2011*, St.Louis, MO, USA, 2011.

[18] B. Khaldi and F. Cherif, "An Overview of Swarm Robotics : Swarm Intelligence Applied to Multi-robotics," *International Journal of Computer Applications*, vol. 126, no. 2, pp. 31–37, 2015.

[19] C. E. Pippin and H. Christensen, "Cooperation based dynamic team formation in multi-agent auctions," in *Proc. of SPIE*, vol. 8389, 2012, pp. 838 919–838 919–8.

[20] C. M. Clark, R. Morton, and G. A. Bekey, "Altruistic relationships for optimizing task fulfillment in robot communities," *Distributed Autonomous Robotic Systems 8*, pp. 261–270, 2009.

[21] R. D. Morton, G. A. Bekey, and C. M. Clark, "Altruistic task allocation despite unbalanced relationships within Multi-Robot Communities," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, St.Louis, MO, USA, oct 2009, pp. 5849–5854.

[22] Robotics Trends, "4 Challenges Holding Back Swarm Robotics," 2015. [Online]. Available: http://www.roboticstrends.com/article/4_challenges_holding_back_swarm_robotics

[23] M. Chamanbaz, D. Mateo, B. M. Zoss, G. Tokic, E. Wilhelm, R. Bouffanais, and D. K. P. Yue, "Swarm-Enabling Technology for Multi-Robot Systems," *Frontiers in Robotics and AI*, vol. 4, pp. 1–18, 2017.

[24] B. P. Gerkey and M. J. Matarić, "A Formal Analysis and Taxonomy of Task Allocation in Multi-robot Systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[25] G. A. Korsah, A. Stentz, and M. B. Dias, "A Comprehensive Taxonomy for Multi-robot Task Allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[26] O. Shehory and S. Kraus, "Methods for Task Allocation via Agent Coalition Formation," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 165–200, 1998.

[27] R. B. Myerson, *Game Theory: Analysis of Conflict*. Cambridge, Massachusetts, USA: Harvard Univercity Press, 1997.

[28] A. Darmann, "Group Activity Selection from Ordinal Preferences," in *Algorithmic Decision Theory. ADT 2015. Lecture Notes in Computer Science*, ser. Lecture Notes in Computer Science, T. Walsh, Ed. Berlin, Heidelberg: Springer Cham, 2015, vol. 9346, pp. 35–51.

References for Chapter 2

[1] E. Sahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 10–20.

[2] B. Acikmese and D. S. Bayard, "A Markov Chain Approach to Probabilistic Swarm Guidance," in *American Control Conference*, Montreal, Canada, 2012, pp. 6300–6307.

[3] B. Acikmese and D. S. Bayard, "Probabilistic Swarm Guidance for Collaborative Autonomous Agents," in *American Control Conference*, Portland, OR, USA, 2014, pp. 477–482.

[4] B. Acikmese and D. S. Bayard, "Markov Chain Approach to Probabilistic Guidance for Swarms of Autonomous Agents," *Asian Journal of Control*, vol. 17, no. 4, pp. 1105–1124, 2015.

[5] I. Chattopadhyay and A. Ray, "Supervised Self-Organization of Homogeneous Swarms Using Ergodic Projections of Markov Chains," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1505–1515, 2009.

[6] N. Demir and B. Acikmese, "Probabilistic Density Control for Swarm of Decentralized ON-OFF Agents with Safety Constraints," in *American Control Conference*, Chicago, IL, USA, 2015, pp. 5238–5244.

[7] R. Luo, N. Chakraborty, and K. Sycara, "Supervisory Control for Cost-Effective Redistribution of Robotic Swarms," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, San Diego, CA, USA, 2014, pp. 596–601.

[8] N. Demir, U. Eren, and B. AçÄśkmeÅ§e, "Decentralized Probabilistic Density Control of Autonomous Swarms with Safety Constraints," *Autonomous Robots*, vol. 39, no. 4, pp. 537–554, 2015.

[9] D. Morgan, G. P. Subramanian, S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic Guidance of Distributed Systems using Sequential Convex Programming," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA, 2014, pp. 3850–3857.

[10] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1103–1123, 2017.

[11] A. Halasz, M. A. Hsieh, S. Berman, and V. Kumar, "Dynamic Redistribution of a Swarm of Robots among Multiple Sites," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, 2007, pp. 2320–2325.

[12] M. A. Hsieh, A. Halasz, S. Berman, and V. Kumar, "Biologically Inspired Redistribution of a Swarm of Robots among Multiple Sites," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 121–141, 2008.

[13] A. Prorok, M. A. Hsieh, and V. Kumar, "The Impact of Diversity on Optimal Control Policies for Heterogeneous Robot Swarms," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 346–358, 2017.

[14] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Inhomogeneous Markov Chain Approach To Probablistic Swarm Guidance Algorithms," in *5th Int. Conf. on Spacecraft Formation Flying Missions and Technologies (SFFMT)*, Munich, Germany, 2013.

[15] K. Lerman, A. Martinoli, and A. Galstyan, "A Review of Probabilistic Macroscopic Models for Swarm Robotic Systems," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 143–152.

[16] T. W. Mather and M. A. Hsieh, "Macroscopic Modeling of Stochastic Deployment Policies with Time Delays for Robot Ensembles," *The International Journal of Robotics Research*, vol. 30, no. 5, pp. 590–600, 2011.

[17] S. Berman, A. Halasz, M. Ani Hsieh, and V. Kumar, "Navigation-based Optimization of Stochastic Strategies for Allocating a Robot Swarm among Multiple Sites," in *IEEE Conference on Decision and Control*, Cancun, Mexico, 2008, pp. 4376–4381.

[18] S. Berman, A. Halasz, M. A. Hsieh, and V. Kumar, "Optimized Stochastic Policies for Task Allocation in Swarms of Robots," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 927–937, 2009.

[19] L. B. Johnson, H.-L. Choi, and J. P. How, "The Role of Information Assumptions in Decentralized Task Allocation: A Tutorial," *IEEE Control Systems*, vol. 36, no. 4, pp. 45–58, 2016.

[20] I. D. Couzin, J. Krause, R. James, G. D. Ruxton, and N. R. Franks, "Collective Memory and Spatial Sorting in Animal Groups." *Journal of theoretical biology*, vol. 218, no. 1, pp. 1–11, 2002.

[21] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective Leadership and Decision-making in Animal Groups on the Move," *Nature*, vol. 433, no. 7025, pp. 513–516, 2005.

[22] J. Gautrais, C. Jost, and G. Theraulaz, "Key Behavioural Factors in a Self-Organised Fish School Model," *Annales Zoologici Fennici*, vol. 45, no. 5, pp. 415–428, 2008.

[23] D. J. Hoare, I. D. Couzin, J. G. J. Godin, and J. Krause, "Context-dependent Group Size Choice in Fish," *Animal Behaviour*, vol. 67, no. 1, pp. 155–164, 2004.

[24] T. D. Seeley, *The Wisdom of the Hive*. Cambridge, Massachusetts: Havard University Press, 1995.

[25] L. Keller, M. J. B. Krieger, and J.-B. Billeter, "Ant-like Task Allocation and Recruitment in Cooperative Robots," *Nature*, vol. 406, no. 6799, pp. 992–995, 2000.

[26] B. L. Partridge, "The Structure and Function of Fish Schools," *Scientific American*, vol. 246, no. 6, pp. 114–123, 1982.

[27] C. Becco, N. Vandewalle, J. Delcourt, and P. Poncin, "Experimental Evidences of a Structural and Dynamical Transition in Fish School," *Physica A: Statistical Mechanics and its Applications*, vol. 367, pp. 487–493, 2006.

[28] S. Bandyopadhyay and S.-J. Chung, "Distributed Estimation using Bayesian Consensus Filtering," *American Control Conference*, pp. 634–641, 2014.

[29] Y. Bestaoui Sebbane, *Planning and Decision Making for Aerial Robots*, ser. Intelligent Systems, Control and Automation: Science and Engineering. Cham: Springer International Publishing, 2014, vol. 71.

[30] I. C. F. Ipsen and T. M. Selee, "Ergodicity Coefficients Defined by Vector Norms," *SIAM Journal on Matrix Analysis and Applications*, vol. 32, no. 1, pp. 153–200, jan 2011.

[31] E. Seneta, *Non-negative Matrices and Markov Chains*, ser. Springer Series in Statistics. Springer New York, 1981.

[32] L. Johnson, S. Ponda, H.-L. Choi, and J. How, "Asynchronous Decentralized Task Allocation for Dynamic Environments," in *Infotech@Aerospace*, St.Louis, MO, USA, 2011.

[33] A. Jadbabaie, Jie Lin, and A. Morse, "Coordination of Groups of Mobile Autonomous Agents using Nearest Neighbor Rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[34] J. Chung, P. Kannappan, C. Ng, and P. Sahoo, "Measures of Distance between Probability Distributions," *Journal of Mathematical Analysis and Applications*, vol. 138, no. 1, pp. 280–292, 1989.

[35] S. A. M. Saleh, S. A. Suandi, and H. Ibrahim, "Recent Survey on Crowd Density Estimation and Counting for Visual Surveillance," *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 103–114, 2015.

[36] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. Cambridge: Cambridge University Press, 2012.

References for Chapter 3

[1] J. H. Drèze and J. Greenberg, "Hedonic Coalitions: Optimality and Stability," *Econometrica*, vol. 48, no. 4, pp. 987–1003, 1980.

[2] S. Banerjee, H. Konishi, and T. Sönmez, "Core in a Simple Coalition Formation Game," *Social Choice and Welfare*, vol. 18, no. 1, pp. 135–153, 2001.

[3] A. Bogomolnaia and M. O. Jackson, "The Stability of Hedonic Coalition Structures," *Games and Economic Behavior*, vol. 38, no. 2, pp. 201–230, 2002.

[4] B. P. Gerkey and M. J. Matarić, "A Formal Analysis and Taxonomy of Task Allocation in Multi-robot Systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[5] G. A. Korsah, A. Stentz, and M. B. Dias, "A Comprehensive Taxonomy for Multi-robot Task Allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[6] A. Brutschy, G. Pini, C. Pinciroli, M. Birattari, and M. Dorigo, "Self-organized Task Allocation to Sequentially Interdependent Tasks in Swarm Robotics," *Autonomous Agents and Multi-Agent Systems*, vol. 28, no. 1, pp. 101–125, 2014.

[7] N. Kalra and A. Martinoli, "A Comparative Study of Market-Based and Threshold-Based Task Allocation," in *Distributed Autonomous Robotic Systems 7*, Tokyo, Ed. Springer Japan, 2006, pp. 91–101.

[8] Y. Zhang and L. E. Parker, "Considering Inter-task Resource Constraints in Task Allocation," *Autonomous Agents and Multi-Agent Systems*, vol. 26, no. 3, pp. 389–419, 2013.

[9] H. L. Choi, L. Brunet, and J. P. How, "Consensus-based Decentralized Auctions for Robust Task Allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[10] P. Segui-Gasco, H.-S. Shin, A. Tsourdos, and V. J. Segui, "Decentralised Submodular Multi-Robot Task Allocation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hamburg, Germany, 2015, pp. 2829–2834.

[11] B. Acikmese and D. S. Bayard, "Markov Chain Approach to Probabilistic Guidance for Swarms of Autonomous Agents," *Asian Journal of Control*, vol. 17, no. 4, pp. 1105–1124, 2015.

[12] I. Chattopadhyay and A. Ray, "Supervised Self-Organization of Homogeneous Swarms Using Ergodic Projections of Markov Chains," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1505–1515, 2009.

[13] N. Demir and B. Acikmese, "Probabilistic Density Control for Swarm of Decentralized ON-OFF Agents with Safety Constraints," in *American Control Conference*, Chicago, IL, USA, 2015, pp. 5238–5244.

[14] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1103–1123, 2017.

[15] I. Jang, H.-S. Shin, and A. Tsourdos, "Bio-Inspired Local Information-Based Control for Probabilistic Swarm Distribution Guidance," *arXiv:1711.06869 [cs.MA]*, 2017.

[16] S. Berman, A. Halasz, M. A. Hsieh, and V. Kumar, "Optimized Stochastic Policies for Task Allocation in Swarms of Robots," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 927–937, 2009.

[17] A. Halasz, M. A. Hsieh, S. Berman, and V. Kumar, "Dynamic Redistribution of a Swarm of Robots among Multiple Sites," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, 2007, pp. 2320–2325.

[18] M. A. Hsieh, A. Halasz, S. Berman, and V. Kumar, "Biologically Inspired Redistribution of a Swarm of Robots among Multiple Sites," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 121–141, 2008.

[19] T. W. Mather and M. A. Hsieh, "Macroscopic Modeling of Stochastic Deployment Policies with Time Delays for Robot Ensembles," *The International Journal of Robotics Research*, vol. 30, no. 5, pp. 590–600, 2011.

[20] A. Prorok, M. A. Hsieh, and V. Kumar, "The Impact of Diversity on Optimal Control Policies for Heterogeneous Robot Swarms," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 346–358, 2017.

[21] T. H. Labella, M. Dorigo, and J.-L. Deneubourg, "Division of Labor in a Group of Robots Inspired by Ants' Foraging Behavior," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 1, no. 1, pp. 4–25, 2006.

[22] E. Castello, T. Yamamoto, Y. Nakamura, and H. Ishiguro, "Foraging Optimization in Swarm Robotic Systems Based on an Adaptive Response Threshold Model," *Advanced Robotics*, vol. 28, no. 20, pp. 1343–1356, 2014.

[23] H. Kurdi, J. How, and G. Bautista, "Bio-Inspired Algorithm for Task Allocation in Multi-UAV Search and Rescue Missions," in *AIAA Guidance, Navigation, and Control Conference*, San Diego, CA, USA, 2016.

[24] W. Liu, A. F. T. Winfield, J. Sa, J. Chen, and L. Dou, "Towards Energy Optimization: Emergent Task Allocation in a Swarm of Foraging Robots," *Adaptive Behavior*, vol. 15, no. 3, pp. 289–305, 2007.

[25] W. Liu and A. F. T. Winfield, "Modeling and Optimization of Adaptive Foraging in Swarm Robotic Systems," *The International Journal of Robotics Research*, vol. 29, no. 14, pp. 1743–1760, 2010.

[26] A. Martinoli, K. Easton, and W. Agassounon, "Modeling Swarm Robotic Systems: a Case Study in Collaborative Distributed Manipulation," *The International Journal of Robotics Research*, vol. 23, no. 4, pp. 415–436, 2004.

[27] K. Lerman, A. Martinoli, and A. Galstyan, "A Review of Probabilistic Macroscopic Models for Swarm Robotic Systems," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 143–152.

[28] N. Correll and A. Martinoli, "System Identification of Self-Organizing Robotic Swarms," in *Distributed Autonomous Robotic Systems 7*. Tokyo: Springer Japan, 2006, pp. 31–40.

[29] A. Prorok, N. Correll, and A. Martinoli, "Multi-level Spatial Modeling for Stochastic Distributed Robotic Systems," *The International Journal of Robotics Research*, vol. 30, no. 5, pp. 574–589, 2011.

[30] A. Kanakia, J. Klingner, and N. Correll, "A Response Threshold Sigmoid Function Model for Swarm Robot Collaboration," *Distributed Autonomous Robotic Systems*, pp. 193–206, 2016.

[31] D. Dimitrov and S. C. Sung, "Top responstiveness and Nash stability in coalition formation games," *Kybernetika*, vol. 42, no. 4, pp. 453–460, 2006.

[32] A. Darmann, E. Elkind, S. Kurz, J. Lang, J. Schauer, and G. Woeginger, "Group Activity Selection Problem," in *Proceedings of the 8th International Conference on Internet and Network Economics*, Liverpool, UK, 2012, pp. 156–169.

[33] A. Darmann, "Group Activity Selection from Ordinal Preferences," in *Algorithmic Decision Theory. ADT 2015. Lecture Notes in Computer Science*, ser. Lecture Notes in Computer Science, T. Walsh, Ed. Berlin, Heidelberg: Springer Cham, 2015, vol. 9346, pp. 35–51.

[34] E. Sahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 10–20.

[35] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable Self-assembly in a Thousand-robot Swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[36] S. C. Sung and D. Dimitrov, "On Myopic Stability Concepts for Hedonic Games," *Theory and Decision*, vol. 62, no. 1, pp. 31–45, 2007.

[37] M. Karakaya, "Hedonic Coalition Formation Games: A New Stability Notion," *Mathematical Social Sciences*, vol. 61, no. 3, pp. 157–165, 2011.

[38] H. Aziz and F. Brandl, "Existence of Stability in Hedonic Coalition Formation Games," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, Valencia, Spain, 2012, pp. 763–770.

[39] J. Guerrero and G. Oliver, "Multi-robot Coalition Formation in Real-time Scenarios," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1295–1307, 2012.

[40] O. Shehory and S. Kraus, "Feasible Formation of Coalitions Among Autonomous Agents in Nonsuperadditive Environments," *Computational Intelligence*, vol. 15, no. 3, pp. 218–251, 1999.

[41] C. Nam and D. A. Shell, "Assignment Algorithms for Modeling Resource Contention in Multirobot Task Allocation," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 889–900, 2015.

[42] L. B. Johnson, H.-L. Choi, and J. P. How, "The Role of Information Assumptions in Decentralized Task Allocation: A Tutorial," *IEEE Control Systems*, vol. 36, no. 4, pp. 45–58, 2016.

[43] W. Saad, Z. Han, M. Debbah, and A. Hjørungnes, "A Distributed Coalition Formation Framework for Fair User Cooperation in Wireless Networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 9, pp. 4580–4593, 2009.

[44] W. Saad, Z. Han, T. Basar, M. Debbah, and A. Hjørungnes, "Hedonic Coalition Formation for Distributed Task Allocation among Wireless Agents," *IEEE Transactions on Mobile Computing*, vol. 10, no. 9, pp. 1327–1344, 2011.

[45] R. E. Allen, A. A. Clark, J. A. Starek, and M. Pavone, "A Machine Learning Approach for Real-Time Reachability Analysis Ross," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA, 2014, pp. 2202–2208.

[46] M. Abdelkader, H. Jaleel, and J. S. Shamma, "A Distributed Framework for Real Time Path Planning in Practical Multi-agent Systems," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 10 626–10 631, 2017.

[47] M. Otte and N. Correll, "Dynamic teams of robots as ad hoc distributed computers: reducing the complexity of multi-robot motion planning via subspace selection," *Autonomous Robots*, 2018.

References for Chapter 4

[1] I. Jang, H.-S. Shin, and A. Tsourdos, "Anonymous Hedonic Game for Task Allocation in a Large-scale Multiple Agent System," *IEEE Transactions on Robotics (in press)*, 2018.

[2] I. Jang, H.-S. Shin, and A. Tsourdos, "Bio-Inspired Local Information-Based Control for Probabilistic Swarm Distribution Guidance," *arXiv:1711.06869 [cs.MA]*, 2017.

[3] A. Brutschy, G. Pini, C. Pinciroli, M. Birattari, and M. Dorigo, "Self-organized Task Allocation to Sequentially Interdependent Tasks in Swarm Robotics," *Autonomous Agents and Multi-Agent Systems*, vol. 28, no. 1, pp. 101–125, 2014.

[4] N. Kalra and A. Martinoli, "A Comparative Study of Market-Based and Threshold-Based Task Allocation," in *Distributed Autonomous Robotic Systems 7*, Tokyo, Ed. Springer Japan, 2006, pp. 91–101.

[5] B. Acikmese and D. S. Bayard, "A Markov Chain Approach to Probabilistic Swarm Guidance," in *American Control Conference*, Montreal, Canada, 2012, pp. 6300–6307.

[6] P. Zahadat and T. Schmickl, "Division of labor in a swarm of autonomous underwater robots by improved partitioning social inhibition," *Adaptive Behavior*, vol. 24, no. 2, pp. 1–11, 2016.

[7] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1103–1123, 2017.

[8] A. Darmann, E. Elkind, S. Kurz, J. Lang, J. Schauer, and G. Woeginger, "Group Activity Selection Problem," in *Proceedings of the 8th International Conference on Internet and Network Economics*, Liverpool, UK, 2012, pp. 156–169.

[9] A. Darmann, "Group Activity Selection from Ordinal Preferences," in *Algorithmic Decision Theory. ADT 2015. Lecture Notes in Computer Science*, ser. Lecture Notes in Computer Science, T. Walsh, Ed. Berlin, Heidelberg: Springer Cham, 2015, vol. 9346, pp. 35–51.

[10] I. Ipsen, "Coefficients of Ergodicity: An Introduction," in *Numerical Analysis Seminar, NC State University (April 23, 2008)*, 2008.

[11] S. Berman, A. Halasz, M. A. Hsieh, and V. Kumar, "Optimized Stochastic Policies for Task Allocation in Swarms of Robots," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 927–937, 2009.

[12] F. Arvin, S. Watson, A. E. Turgut, J. Espinosa, T. Krajník, and B. Lennox, "Perpetual Robot Swarm: Long-Term Autonomy of Mobile Robots Using On-the-fly Inductive Charging," *Journal of Intelligent and Robotic Systems: Theory and Applications*, pp. 1–18, 2017.

[13] A. B. Junaid, Y. Lee, and Y. Kim, "Design and implementation of autonomous wireless charging station for rotary-wing UAVs," *Aerospace Science and Technology*, vol. 54, pp. 253–266, 2016.

[14] I. Jang, J. Jeong, H.-S. Shin, S. Kim, A. Tsourdos, and J. Suk, "Cooperative Control for a Flight Array of UAVs and an Application in Radar Jamming," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8011–8018, 2017.

[15] A. Jevtić, Á. Gutierrez, D. Andina, and M. Jamshidi, "Distributed Bees Algorithm for Task Allocation in Swarm of Robots," *IEEE Systems Journal*, vol. 6, no. 2, pp. 296–304, 2012.

References for Chapter 5

[1] H.-S. Shin and P. Segui-Gasco, "UAV Swarms: Decision-Making Paradigms," in *Encyclopedia of Aerospace Engineering*. Chichester, UK: John Wiley & Sons, Ltd, dec 2014, pp. 1–13.

[2] K. Barton and D. Kingston, "Systematic Surveillance for UAVs: A Feedforward Iterative Learning Control Approach," in *American Control Conference*, Washington, DC, USA, 2013, pp. 5917–5922.

[3] I. Bekmezci, O. K. Sahingoz, and S. Temel, "Flying Ad-Hoc Networks (FANETs): A Survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[4] I. Jang, J. Jeong, H.-S. Shin, S. Kim, A. Tsourdos, and J. Suk, "Cooperative Control for a Flight Array of UAVs and an Application in Radar Jamming," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8011–8018, 2017.

[5] B. P. Gerkey and M. J. Matarić, "A Formal Analysis and Taxonomy of Task Allocation in Multi-robot Systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[6] T. Öncan, "A Survey of the Generalized Assignment Problem and Its Applications," *INFOR: Information Systems and Operational Research*, vol. 45, no. 3, pp. 123–141, 2007.

[7] T. William Mather and M. Ani Hsieh, "Distributed Robot Ensemble Control for Deployment to Multiple Sites," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, 2011.

[8] P. Segui-Gasco, H.-S. Shin, A. Tsourdos, and V. J. Segui, "Decentralised Submodular Multi-Robot Task Allocation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hamburg, Germany, 2015, pp. 2829–2834.

[9] H. L. Choi, L. Brunet, and J. P. How, "Consensus-based Decentralized Auctions for Robust Task Allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[10] L. Luo, N. Chakraborty, and K. Sycara, "Distributed Algorithms for Multirobot Task Assignment With Task Deadline Constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 876–888, 2015.

[11] L. Luo, N. Chakraborty, and K. Sycara, "Provably-Good Distributed Algorithm for Constrained Multi-Robot Task Assignment for Grouped Tasks," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 19–30, 2015.

[12] M. M. Güntzer and D. Jungnickel, "Approximate Minimization Algorithms for the 0/1 Knapsack and Subset-Sum Problem," *Operations Research Letters*, vol. 26, no. 2, pp. 55–66, 2000.

[13] X. Han and K. Makino, "Online Minimization Knapsack Problem," in *Approximation and Online Algorithms. WAOA 2009. Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 2010, vol. 5893, pp. 182–193.

[14] M. J. Geiger and W. Wenger, "On the Assignment of Students to Topics: A Variable Neighborhood Search Approach," *Socio-Economic Planning Sciences*, vol. 44, no. 1, pp. 25–34, 2010.

[15] M. Bender, C. Thielen, and S. Westphal, "A Constant Factor Approximation for the Generalized Assignment Problem with Minimum Quantities and Unit Size Items," in *Mathematical Foundations of Computer Science 2013. MFCS 2013. Lecture Notes in Computer Science, vol 8087*. Berlin: Springer, 2013, pp. 135–145.

[16] S. O. Krumke and C. Thielen, "The Generalized Assignment Problem with Minimum Quantities," *European Journal of Operational Research*, vol. 228, no. 1, pp. 46–55, 2013.

[17] Z. Han, D. Niyato, W. Saad, T. Basar, and A. Hjørungnes, *Game Theory in Wireless and Communication Networks : Theory, Models, and Applications*. Cambridge University Press, 2012.

[18] I. Jang, H.-S. Shin, and A. Tsourdos, "Anonymous Hedonic Game for Task Allocation in a Large-scale Multiple Agent System," *IEEE Transactions on Robotics (in press)*, 2018.

[19] D. Beyer and R. Ogier, "Tabu Learning: A Neural Network Search Method for Solving Nonconvex Optimization Problems," in *IEEE International Joint Conference on Neural Networks*, 1991, pp. 953–961.

[20] N. N. Galim'yanova, A. A. Korbut, and I. K. Sigal, "Ratios of Optimal Values of Objective Functions of the Knapsack Problem and Its Linear Relaxation," *Journal of Computer and Systems Sciences International*, vol. 48, no. 6, pp. 906–913, 2009.

References for Chapter 6

[1] E. Sahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 10–20.

[2] I. Navarro and F. Matía, "An Introduction to Swarm Robotics," *ISRN Robotics*, vol. 2013, pp. 1–10, 2013.

[3] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm Robotics: a Review from the Swarm Engineering Perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[4] K. Barton and D. Kingston, "Systematic Surveillance for UAVs: A Feedforward Iterative Learning Control Approach," in *American Control Conference*, Washington, DC, USA, 2013, pp. 5917–5922.

[5] I. Bekmezci, O. K. Sahingoz, and S. Temel, "Flying Ad-Hoc Networks (FANETs): A Survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[6] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz, "Help from the Sky: Leveraging UAVs for Disaster Management," *IEEE Pervasive Computing*, vol. 16, no. 1, pp. 24–32, 2017.

[7] I. Jang, J. Jeong, H.-S. Shin, S. Kim, A. Tsourdos, and J. Suk, "Cooperative Control for a Flight Array of UAVs and an Application in Radar Jamming," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8011–8018, 2017.

[8] B. P. Gerkey and M. J. Matarić, "A Formal Analysis and Taxonomy of Task Allocation in Multi-robot Systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[9] G. A. Korsah, A. Stentz, and M. B. Dias, "A Comprehensive Taxonomy for Multi-robot Task Allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[10] H.-S. Shin and P. Segui-Gasco, "UAV Swarms: Decision-Making Paradigms," in *Encyclopedia of Aerospace Engineering*. John Wiley & Sons, 2014, pp. 1–13.

[11] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica*, vol. 33, no. 03, pp. 463–497, 2015.

[12] B. Saicharan, R. Tiwari, and N. Roberts, "Multi Objective optimization based Path Planning in robotics using nature inspired algorithms: A survey," in *IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems*, Delhi, India, 2016, pp. 1–6.

[13] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable Self-assembly in a Thousand-robot Swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[14] O. Shehory and S. Kraus, "Methods for Task Allocation via Agent Coalition Formation," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 165–200, 1998.

[15] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Image and animation display with multiple mobile robots," *International Journal of Robotics Research*, vol. 31, no. 6, pp. 753–773, 2012.

[16] M. Turpin, N. Michael, and V. Kumar, "CAPT: Concurrent assignment and planning of trajectories for multiple robots," *International Journal of Robotics Research*, vol. 33, no. 1, pp. 98–112, 2014.

[17] M. Turpin, K. Mohta, N. Michael, and V. Kumar, "Goal assignment and trajectory planning for large teams of interchangeable robots," *Autonomous Robots*, vol. 37, no. 4, pp. 401–415, 2014.

[18] D. Morgan, G. P. Subramanian, S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic Guidance of Distributed Systems using Sequential Convex Programming," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA, 2014, pp. 3850–3857.

[19] D. Morgan, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, "Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1261–1285, 2016.

[20] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, "Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 6, pp. 1725–1740, 2014.

[21] J. D.F. Votaw and A. Orden, "The Personnel Assignment Problem," in *Symposium on Linear Inequalities and Programming*, Washington, DC: Planning Research Division, Comptroller, Headquarters US Air Force, 1952, pp. 155–163.

[22] I. Jang, H.-S. Shin, and A. Tsourdos, "A Game-theoretical Approach to Heterogeneous Multi-robot Task Assignment Problem with Minimum Workload Requirements," in *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, Linkoping, Sweden, 2017, pp. 156–161.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithm*, 3rd ed. Cambridge, Massachusetts, USA: The MIT Press, 2009.

[24] M. M. Güntzer and D. Jungnickel, "Approximate Minimization Algorithms for the 0/1 Knapsack and Subset-Sum Problem," *Operations Research Letters*, vol. 26, no. 2, pp. 55–66, 2000.

[25] D. Beyer and R. Ogier, "Tabu Learning: A Neural Network Search Method for Solving Nonconvex Optimization Problems," in *IEEE International Joint Conference on Neural Networks*, 1991, pp. 953–961.

[26] I. Jang, H.-S. Shin, and A. Tsourdos, "Anonymous Hedonic Game for Task Allocation in a Large-scale Multiple Agent System," *IEEE Transactions on Robotics (conditionally accepted)*, 2018.

[27] H. W. Khun, "The Hungarian Method for the Assignment Problem," *Naval Reserach Logistic Quarterly*, vol. 2, pp. 83–97, 1955.

[28] K. Mehlhorn and P. Sanders, *Algorithms and Data Structures*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[29] Y. Qin, M. Cao, and B. D. O. Anderson, "Asynchronous Agreement through Distributed Coordination Algorithms Associated with Periodic Matrices," in *Proceedings of the IFAC World Congress*, Toulouse, France, 2017, pp. 1778–1783.

[30] L. B. Johnson, H.-L. Choi, and J. P. How, "The Role of Information Assumptions in Decentralized Task Allocation: A Tutorial," *IEEE Control Systems*, vol. 36, no. 4, pp. 45–58, 2016.

[31] J. Kim and P. Hespanha, "Cooperative radar jamming for groups of unmanned air vehicles," in *IEEE Conference on Decision and Control*, Atlantis, Bahamas, 2004, pp. 632–637.

[32] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, ver 2.1," 2017.

References for Chapter 7

[1] E. Sahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," in *Swarm Robotics*. Berlin: Springer, 2005, pp. 10–20.

[2] M. Dorigo, M. Birattari, and M. Brambilla, "Swarm robotics," *Scholarpedia*, vol. 9, no. 1, p. 1463, 2014.

[3] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1103–1123, 2017.

[4] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm Robotics: a Review from the Swarm Engineering Perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[5] L. Johnson, S. Ponda, H.-L. Choi, and J. How, "Asynchronous Decentralized Task Allocation for Dynamic Environments," in *Infotech@Aerospace*, St.Louis, MO, USA, 2011.

[6] B. L. Partridge, "The Structure and Function of Fish Schools," *Scientific American*, vol. 246, no. 6, pp. 114–123, 1982.

[7] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective Leadership and Decision-making in Animal Groups on the Move," *Nature*, vol. 433, no. 7025, pp. 513–516, 2005.

[8] C. Becco, N. Vandewalle, J. Delcourt, and P. Poncin, "Experimental Evidences of a Structural and Dynamical Transition in Fish School," *Physica A: Statistical Mechanics and its Applications*, vol. 367, pp. 487–493, 2006.

[9] J. Gautrais, C. Jost, and G. Theraulaz, "Key Behavioural Factors in a Self-Organised Fish School Model," *Annales Zoologici Fennici*, vol. 45, no. 5, pp. 415–428, 2008.

[10] A. Darmann, "Group Activity Selection from Ordinal Preferences," in *Algorithmic Decision Theory. ADT 2015. Lecture Notes in Computer Science*, ser. Lecture Notes in Computer Science, T. Walsh, Ed. Berlin, Heidelberg: Springer Cham, 2015, vol. 9346, pp. 35–51.

[11] H.-S. Shin, I. Jang, and A. Tsourdos, "Frequency Channel Assignment for Networked UAVs using a Hedonic Game," in *RED-UAS 2017*, Linkoping, Sweden, 2017.

[12] S. A. M. Saleh, S. A. Suandi, and H. Ibrahim, "Recent Survey on Crowd Density Estimation and Counting for Visual Surveillance," *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 103–114, 2015.