INF FAKULTÄT FÜR INFORMATIK

Otto-von-Guericke-Universität Magdeburg

Faculty of Computer Science

Master Thesis

Discrete Collective Estimation with Different Majority Voting Algorithms in Swarm Robotics

Author:

Alexander Heck

Supervisor:

Prof. Sanasz Mostaghim

Advisor:

Qihao Shan

Magdeburg, May 27, 2021

Abstract

Majority voting systems are widely used in political elections across the globe. In this paper, we have implemented three such strategies as alternatives to collective decision-making to solve the collective perception problem: the first-past-the-post (FPTP), single transferable vote (STV), and Borda count (BC) voting systems. We tested them in different environments with different frequencies of colored tiles, various lengths for the exploration and dissemination state, two feature patterns, and several mean widths for the blocks in the block pattern. We compared the collected data with two common collective decision-making strategies, direct modulation of voter-based decisions (DMVD) and independent decision-making (INDI). We recorded several trends and trade-offs in terms of reliability, accuracy, and convergence time and a strong impact on all algorithms for the block pattern. Our study revealed that borda count shows the best performance and the highest robustness, followed by single transferable vote and first-past-the-post. First-past-the-post is either reliable or accurate, depending on the environment. However, we found that the strong impact of the block pattern renders the performance of Borda count and single transferable vote more vulnerable to feature clustering than that of first-past-the-post.

Contents

List of Abbreviations

Abbreviations for used terms

FPTP	First-past-the-post voting
STV	Single transferable vote
BC	Borda count
INDI	Independent decision-making (INDI)
DMVD	Direct modulation of voter-based decisions
DBHT	Distributed Bayesian Hypothesis Testing
DMMD	Direct modulation of majority-based decisions
BTR	Black-tile-ratio
TI	Time interval
MBW	Mean-block-width
RR	Reliability rate
AE	Absolute error
CT	Convergence time
POI	Point of interest
DM	Decision-making

List of Figures

1 Introduction and Motivation

1.1 Swarm Intelligence and Collective Decision-making

During the last century, numerous scholars have studied the topic of swarm intelligence and the subordinate field of collective decision-making, focusing, in particular, on the problems of site-selection and the collective perception problem, such as Valentini et al. [30],[28], Morlino et al. [14], Strobel et al. [26], Ebert et al. [9], and Shan et al. [25]. The expression "swarm intelligence" was first used by J. Wang and G. Beni in 1989 [31]. Swarm intelligence is a sub-discipline in the field of artificial intelligence that deals with the collective behavior of self-organized systems of robots. Collective decision-making refers to when multiple individuals are faced with making a choice collectively without using a central leader. In this scenario, the decision collectively made by the swarm is not traceable to individual agents. There have been several approaches to dealing with the collective decision problem as explained further by Seok et al. [24]. In our case these individuals are represented by agents who simulate small robots in their behavior. These approaches attempt to reach the best possible performance for the swarm when used to solve different problems. These approaches include the Distributed Bayesian Hypothesis Testing (DBHT) by Shan et al. [25] and Direct modulation of voter-based decisions (DMVD) by Valentini et al. [30]. Most of these decision-making algorithms are based on designs inspired by nature. For example, Valentini et al. [30] observed the nest-searching procedure of honeybees to create strategies to resolve the given problem. These algorithms aim to find the shortest way to a given point of interest (POI) out of two paths with different lengths. This is one of the most well-known best-of-n problems. A more detailed explanation and further examples are given in Chapter 2.

1.2 Contribution of the Thesis and Motivation

This work addresses the collective decision-making and the collective perception problem. Our study uses a square area with a number of tiles, each of which can be either black or white. The task of our swarm is to determine the proportion of the two colors in the arena. This problem was proposed by Valentini et al. [28], who analyzed the generality of the DMVD and the direct modulation of majority-based decisions (DMMD) collective decision-making strategies. We expanded this approach by implementing a "discrete collective estimation" where, the robots select their decision from a discrete list of possible frequencies of the colored tiles, a list of so-called hypotheses. Strobel et al. [7] had their agents choose one of two discrete alternatives, whether the black squares represent more or less than half of the environment. They also used the term "discrete collective swarm estimate". One approach to let the swarm effectively estimate the correct hypotheses is the DBHT approach as proposed by Shan et al. [25]. In this study, we develop three new approaches to create a decision-making strategy with high performance. We exploit an area of general decision-making that has not previously found application in swarm intelligence: majority voting systems. For this purpose, we implemented three algorithms inspired by three different voting system: (1) the first-past-the-post (FPTP) voting, which is currently used by the government of Great Britain; (2) the single transferable vote (STV), which is used widely in Australia; and (3) the Borda count (BC) voting system, which is used for political elections in Slovenia and other countries. Our approaches are unique because each of the agents in the FPTP and STV algorithms forms individual rankings for the possible hypotheses. They determine the quality of the hypotheses in advance and rank them accordingly to continue with the voting. Our approaches consider all hypotheses in each round and not only a set fraction. The goal of our work was to analyze the performance of these algorithms when used for decision-making and to determine if they are a viable alternative as a collective decision-making tool. We tested the algorithms and analyzed the results using reliability, accuracy, and convergence time as metrics. While the reliability measures the capability to converge in time, the accuracy stands for the discrepancy between the decision made by the swarm and the decision that would be correct. The convergence time measures the time taken by the swarm to converge and find the correct decision.

There are several real-world applications for swarm intelligence and, therefore, for collective decision-making. Rosenberg et al. [19] used a human swarm to predict the behavior of financial markets. Cui et al. [6] noted the increasing interest of engineers in swarm intelligence for the fields of robotics, optimization in telecommunication, traffic patterns for transportation systems, and usages in military operations. As shown by Hazem et al. [1], one of the first applications of particle swarm optimization was in the field of bioinformatics. It has been used for applications that include but are not limited to the following: selection of biomarkers, gene clustering, protein structure prediction, and human movement biomechanics [1]. Jevtić et al. [11] highlighted some of the different usages of swarm robotics. The scenarios where swarm robotics and ultimately swarm intelligence and collective decision-making can be applied include foraging, collective transport, the fulfillment of dangerous tasks like demining, exploration, and the mapping of different environments. The various fields of application are especially interesting for our case because a reliable, accurate, and fast decision-making would improve these real-world usages and pave the way for further applications because a higher performance would increase the attractiveness of collective decision-making for the usage on several best-of-n problems and ultimately more diverse problems.

1.3 Structure of the Thesis

After we gave a motivation for our work, we give an overview in Chapter 2 (Related Works). It focuses on the literature related to our field of research, focusing on the origins of swarm intelligence and other approaches to similar problems like our work does. It also details the settings and metrics used in other works and other problems concerning collective decision-making.

Chapter 3 (Methodology) explains our methodology, including used metrics and the test settings, and present the designed majority voting algorithms. We show examples of our algorithms in pseudocode and explain the mechanics of our swarm further, including the states, the movement routine, and the agents themselves. In the end, the reader will have a clear picture of how the majority voting algorithms work, how we implemented them, how we obtained our data, and how we analyzed it.

The results and their subsequent analysis are included in Chapter 4 (Analysis and Results). This chapter explains the expected results, compares them with the actual outcomes, explains why we obtained them, and interprets them in relation to the algorithms and environment. We also provide recommendations for the use of the algorithms and point to existing trade-offs and noticeable trends.

This thesis concludes with Chapter 5 (Conclusion), which summarizes the most important results. It also gives an outlook for future research.

2 Related Works

In this chapter we discuss some works and experiments done by other authors which are related to our topic and to the content of our work. We will give a brief overlook over swarm intelligence and its origins, show some developed alternative strategies and shed light on how other authors have dealt with the issue and what methods they have used for their experiments. At the end of the chapter, we present the underlying strategies for our algorithms to explain them in more detail in Chapter 3.

2.1 Origins and Basic Functionality of Swarm Robotics and Intelligence

Swarm intelligence describes a system consisting of simple agents with a collective behavior. These agents form a so-called swarm which is self-organized and usually decentralized. The agents can only communicate locally. In most experiments done in the real world these agents are represented by simple robots which can communicate via wireless transmission like infrared. These applications are then part of the field swarm robotics.
The term *swarm* is tries to replicate what what we call a swarm in nature. The observation of nature is a common approach that has inspired many swarm intelligence algorithms. There are several social animals that exhibit swarming behavior. Morlino et al. [14], Valentini et al. [30], Passino et al. [16], and others [9] [22] [23] [15] have compared the behavior of honeybees with the swarming robots. These insects also form a decision through a decision-making process based on the knowledge and estimated qualities of each individual. Valentini et al. [30] compared the communication and information exchange inside one swarm of bees with others. Honeybees communicate by the typical

waggle-dance to explain their estimation of possible options to the swarm. One possible decision that needs to be made by the whole swarm would be the process of finding a new home, as Seeley et al. [23] explained. The authors analyzed the entire process of the house-hunting, from the dance to the decision. Bees are not the only animals that use collective decision-making. Other species that exhibit swarm behavior are ants. Valentini et al. [30], Bonabeau et al. [3], Gambardella et al. [10], and Brutschy et al. [5] explained that the approach of swarm intelligence with ants as a role model is called *ant colony optimization*. In contrast to bees, they use pheromones to communicate and ultimately form a decision such as which way to take or which enemy to attack. Algorithms based on ant colony optimization use similar methods to recreate these behaviors for swarm robotics and other fields of research.

Taking into account that bees and ants are social animals, Rosenberg et al. [18] [19] went further and applied the approach of collective decision-making to humans. They introduced UNU, an online platform where participants can take part in a swarm modeled after an artificial swarm intelligence. This swarm was tasked with making decisions and answering questions [18]. In additional experiments, they tasked a swarm of 75 participants formed by UNU with employing the knowledge and prediction skills of football experts and other individuals. By combining their individual skills and knowledge, the swarm was able to beat the predictions of football experts [18].

2.2 Collective Decision-making Problems and Collective Perception Problems

A typical use case for swarm intelligence is the best-of-n decision problem, where a swarm of individuals needs to determine the best decision out of n possible decisions. Valentini et al. [30] [29] [28] and Prasetyo et al. [17] have considered such problems in their experiments. They adopted similar strategies as we do in our experiments. We also have a set of possible decisions, one of which is correct. The task of the swarm is to find that solution. The main goal for creating such an algorithm as explained by Bonabeau et al. [3], is to design a meta-heuristic that works for as many problems as possible.

An artificial intelligent swarm made of semi-intelligent robots is claimed to be fault-

tolerant. Strobel et al. [26] analyzed the behavior of swarms with Byzantine robots, also known as faulty robots or robots with malicious behavior. Under some circumstances, a robot becomes corrupt due to broken sensors, wheels, or other parts. It is also possible that the behavior of a single robot is hacked, and it tries to interrupt decision-making. A hard-drive failure may also lead to missing information that corrupts the robot. In Strobel et al.'s experiments [26], they analyzed the security issues a robot swarm could face and tried to solve them through a blockchain-based approach. They showed that their solution has a significant advantage in terms of security and reliability if the swarm contains Byzantine robots.

To divide the field of collective decision-making, Bonarini et al. [4] have split it into two subclasses of functionality. On the one hand, there is consensus achievement, which is when the swarm converges on a single option from a set of possible decisions. The best-of-n decision is basically the same as consensus achievement. On the other hand, there is the problem of task allocation, which is when robots are asked to spread out over a set of different tasks that need to be accomplished. A typical goal of this subclass is for the agents to distribute themselves so that the performance of the given tasks is maximized. This study will concentrate on decision-making with the consensus achievement as target, and we will not consider any further specialization swarms. We have chosen to focus on this method because it delivers the best possibility to test the majority voting algorithms on a swarm.

2.3 Common Strategies

Decision-making is the key feature of the swarm as well as a critical factor in the convergence for most best-of-n decision problems. It is the part of the algorithm where agents communicate with other agents and obtain a new decision based on the outcome of this communication. The decision-making process is the main part of this thesis. We designed majority voting algorithms and benchmarked them to be comparable to other well-known strategies such as DMVD, DMMD and k-unanimity.

Bonarini et al. [4] provided a classification for different kinds of approaches to decision-making. One example is a centralized system, where one robot represents the central

robot and has knowledge about the entire swarm. Another includes a decentralized system, where no robots are tasked with coordinating the other robots, and no robots have knowledge about the entire swarm. Instead, each robot only has knowledge about the robots close to it and inside the communication radius; movement, decision-making, and all other aspects of swarm behavior rely only on the information provided by the information collected by the swarm.

Most of the decision-making strategies rely on two states that influence the behavior of the agents: the exploration and the dissemination state. These states are used by the approaches of Shan et al. [25], Valentini et al. [30], Parker [15]. Robots in the exploration state distribute in a random walk across the area and estimate the likeliness of the decisions. The dissemination state is the period during which the decision-making takes place. During this state, the robots still walk the area randomly but search for other agents in a dissemination state nearby that are willing to share their decisions. At the end of the state, the robot changes its own decision depending on the used decision-making strategy. The states alternate between each other.

The k-unanimity rule is a decision-making strategy that was introduced by Scheidler et al. [21] and discussed further by Valentini et al. [29]. Scheidlers experiments were based on ant colony optimization and a swarm of robots tasked with finding the shortest path out of two ways to a declared position. The robots resemble the ants, and the position resembles a POI for the swarm of ants, such as a feeding place. The two ways each have a different length. In theory, the agents have two possible decisions. The method enables the swarm of robots to find the decision with the shortest execution time. To achieve this decision, Scheidler et al. [21] proposed the k-unanimity rule. At the start, each robot has a random decision and takes a random path out of the two possible ones. Each robot has a memory that contains the decision of the k latest other robots approached by the robot in dissemination state. If all decisions saved in the memory are the same, the decision of the robot changes according to the dominant decision. This method works because the robots that take the shorter path tend to approach other agents more frequently.

Valentini et al. took the use and the meaning of the time intervals of the dissemination state further with their Direct modulation of voter-based decisions (DMVD) approach [30]. While having a fixed duration of both states, the time the robot spends in dissemination is weighted. It is proportional to the robots' estimated quality of the proposed

decision so that only a decision with the highest possible quality will be adverted the entire dissemination time. Through this procedure, the decision with the better-estimated qualities will be present in more neighborhoods for other agents and thus spread their decision further. DMVD is one of the two decision-making approaches we used to compare to our proposed algorithms.

A different approach to a self-organized decision-making strategy used by Valentini et al. [29] is the DMMD. They implemented the DMMD for an environment with a nest and two sites of interest. This strategy also uses a weighted voter model related to the quality of the proposed decision, similar to DMVD. In contrast to DMVD, DMMD implements individual decisions for agents (robots) and the majority rule. In the majority rule, agents form teams out of a certain number of agents and search for the majority of one of the two decisions. Parker et al. [15] have taken the search for a new nest by honeybees and Temnothorax ants as a role model for creating a decision-making strategy, which they have called *collective comparison*. In their previous work, they determined similarities in the behavior of two different species of insects. Honeybees and Temnothorax ants share similar treats when looking for a new place to rebuild their nest. They send out scouts to find potential places and determine their quality. Afterward, they head back to the current nest and, after a delay, try to recruit a new scout, which then follows them to the new nesting place to determine its quality. If a specimen is recruited by a scout, it immediately forgets the current decision it holds. The key to this decision-making is the delay. The better the nest, the shorter the delay, which means that more potential scouts can be recruited. The collective comparison is inspired by these similar feats and by the communication delay, in particular, based on the quality of the present decision. While Parker et al. [15] have used the term collective comparison, Bonarini et al. [4], Shan et al. [25], and Bartashevich et al. [2] have used the direct name comparison, which will be the term used in this thesis.

A new method for collective perception is called Distributed Bayesian Hypothesis Testing (DBHT) and was proposed by Shan et al. [25]. This method is inspired by sensor fusion techniques which are used in several sensor techniques which are using Bayesian reasoning. It also requires the robots to build up an opinion about the estimated quality of the hypotheses by observing the surrounding environment when wandering the area. The final swarm opinion is built by a leader. The leader is a robot which regularly collects the

opinion from other robots and forms the final swarm estimation.

All decision-making systems are designed for an environment and tested there. The environment can also be called the arena. It is the area where the robots need to fulfill their tasks. In our study, it is the wandering, the observing, and the forming of a consensus through decision-making.

The decision-making strategies of Valentini et al. [30][29], Passino et al. [16], and Scheidler et al. [21] concentrate on an arena with a nest that serves as the headquarters for the robots and POI, which are called sides. There can be only one side and two or more ways with different lengths, such as in the experiment of Scheidler et al. [21]. Another interpretation is that we have a square area split into sections. The nest is the middle section, and the outer sides are the POIs, each with a different quality. Between the nest and the sides, there is space left through which the robots need to pass to reach the POIs. The robots need to find a consensus over which side is better.

There is also another environment setup used, for example, by Shan et al. [25], Bartashevich et al. [2], Morlino et al. [14], Valentini et al. [28], and Strobel et al. [7] [26], which relies on a square as the base. Each side of the square is typically 2 meters long [25] [2] [14] [28], and it is divided into a number of tiles of equal size. The experiments made in [25] [2] [28] are using a square that has been divided into 20 x 20 tiles. Morlino et al. [14] broke down the square into 40 x 40 tiles.

In the experiments of Shan et al. [25] and the others mentioned above, these tiles have two different colors, either white or black. In those experiments, the swarm is tasked with finding out which color is represented more often.

Another approach using a 2D-square as the environment was employed by Ebert et al. [9]. They used different measurements than the others, with each side measuring 2.4 meters. Their algorithm is designed to be used with a multi-feature environment, which, in this case, entails having an arena with more than two colors. The arena they created for their experiments is a combination of three different areas of different colors. Arenas in red, green, and blue are laid on top of each other. The combination of the colors creates a new environment with new colors. Each robot observes one of the base colors at the same time and disseminates the estimates about all colors to its neighbors. The key to this approach is an implemented feature called feature switching. At a specific point in the

algorithm the agent switches the color it is currently estimating the quality for.

There have been additional experiments performed by Bartashevich et al. [2] regarding the environment. In the experiments from the studies previously discussed, the feature of the environment, the black or colored tiles, were distributed randomly in the environment. The performance on a random pattern has been a universal benchmark for many experiments. In the new approach explained by Bartashevich et al. in [2], the feature distribution was changed to create better and more challenging benchmarks for collective decision-making algorithms. Bartashevich et al. [2] created patterns with the tiles of different colors and tested common decision-making strategies in the new environments. Those patterns include, but are not limited to, blocks, stripes, and stars. They tested DMMD, DMVD, and direct comparison (DC) on these new challenges. In this thesis, we make use of the proposed block pattern to analyze and benchmark the designed majority voting algorithms.
All of the mentioned experiments and decision-making strategies rely on the use of agents, which we employ in our work as well. The agents tend to move randomly over the map with a square arena in experiments like in [9], as mentioned before. They do that in exploration and in dissemination state. For algorithms that need a nest and several points of interest, the agents travel back and forth between these POIs on the sides. In the approach of Valentini et al. [29], there are a nest and two sides with different qualities. In this case, the robots use a random walk and an oriented motion pattern between the nest and the POIs to reach the light source, which was installed at the sides. Afterward, the robots return to the nest to begin their dissemination.

These agents can be represented by physical robots as seen by Morlino et al. [14] and Valentini et al [30], [29] or they can be simulated robots as used by Shan et al. [25] and Bartashevich et al. [2]. Having a setup with physical robots means more preparation time and increased expenses. The experiments performed for our benchmarks concentrate on simulated robots, which we refer to as agents later on. A neighborhood is formed during the dissemination state. This neighborhood contains encountered agents which were also in the dissemination state. A agent counts as encountered when it enters the communication radius of the agent. The neighborhood also includes the seeking agent itself. Shan

et al. [25] experimented with this neighborhood and its size. They found that limiting the maximum number of agents in the neighborhood creates trade-offs in decision time, design complexity, and the robustness of the system.

2.4 Voting Systems

Most of the mentioned strategies have been developed based on animals or other examples from nature. For our study, we went deeper into the history of countries and humanity itself. Our proposed algorithms for collective decision-making have been inspired by different ways of democratic voting.

Democracy is a form of government where the people determine their authorities through an election. It is different from other forms of government, such as monarchy or dictatorship, because the power lies within the community and not in bloodlines or fear and suppression. The term democracy first appeared in ancient Greece in the city-states of Athens as explained by J.Dunn [8]. It then started to spread from there. Over the centuries, democracy has evolved into many different forms, such as direct democracy, representative democracy, or its socialist form. However, there is one important detail that all of these forms include: an election and the count of the votes. Today, there are many different ways of counting the votes and distributing them across the candidates. We concentrate on the majority voting systems in this work because they are the basis of our three electoral systems.

The first-past-the-post (FPTP) is the majority voting system in its purest form. During an election, following the path of FPTP, the candidate with the most votes wins. All others lose and are not relevant anymore. Points of criticism include, for example, the number of wasted votes or the fact that smaller parties draw votes from larger, similar parties. This is the role model for our first implemented algorithm.

One of the variants of majority voting strategies is the single transferable vote (STV) explained by N.Tidemann in [27]. It was imagined and created by Thomas Wright Hill in 1819. It is used for different kinds of elections, such as legislative assembly elections and city elections, in many different countries and states around the world; however, it is mostly used by the English-speaking parts of the world. This method is usually used to

fill several seats in a government. The STV is based on a principle where every person is allowed to vote once but has a number of choices. These choices resemble the ranks of the candidates for the voter. His or her first choice is the favorite, the next their second choice, and so on. In the first round, the vote goes to the first choice. If that candidate manages to gain a certain percentage of votes, they become part of the elected government. At the end of round two, the lowest-performing participant is kicked out, and their votes go to the next highest-ranking candidate. This process continues until all seats are full [27]. One of our proposed majority voting algorithms is based on the STV. The last method is inspired by the so-called Borda count (BC), explained by Russell [20]. It is an election method for a single winner and was first proposed by Nicholas of Cusa in 1435, but it is named after Jean-Charles de Borda, who devised it in 1770. It is currently used in the political elections of Slovenia, Kiribati, and Nauru. BC is based on a scoring system and a ranking similar to that of STV. Each candidate earns as many points as their rank on each of the ballots. In other words, the rank of one receives the best possible number of scores from this voter. The second choice receives one point less and so on. In the end, the candidate with the highest score wins. It was developed to elect widely accepted options rather than those preferred by the majority. That is why some declare it as a consensus-based voting method rather than a majority voting strategy, as mentioned by Lippman [12].

3 Methodology

To achieve test results in terms of benchmarking the decision-making (DM) and the collective decision problem, we used the same environment settings as Shan et al. [25] and Bartashevich et al. [2], which consisted of an arena in the form of a 2x2 meter rectangle and split up into 400 black and white tiles. From this point forward, we always consider the white tiles as the majority in the environment. The black tiles have a certain ratio of the environment, which is fixed as black-tile-ratio of BTR $\{0.05, 0.15, 0.25, 0.35, 0.45\}$. We used a discrete distribution for our experiments. The arena was swarmed by 20 artificial intelligent agents/robots that are able to observe their environment and estimate quality for each provided hypothesis. The hypotheses were given in a list H, which holds the number of estimated ratios of black tiles in the environment. If an agent voted for a hypothesis, it voted for the index of the hypothesis in H. Thus, if an agent currently estimated that the ratio of black tiles in the environment is 0.15, then it voted for the second entry of H; therefore, its current decision is 1 because the second entry has an index 1 in the list. When wandering the arena and applying the collective perception problem, an agent can occupy one of two states. An agent in the exploration state wanders the arena randomly and samples the environment to form estimated qualities for each hypothesis in H. After the exploration state, it changes into the dissemination state. During the dissemination state, an agent also wanders around the arena in the search for other agents that are also in the dissemination state and willing to communicate. The length of the dissemination state is relative to the estimated quality of the actual favored decision. When the quality of the decision that it would use for the DM is better, then the dissemination interval increases by a proportion of the set time interval. With that, we followed Valentini et al. [29] and Shan et al. [25], who used this method for their experiments. For the following equation $t_{diss}(i)$ is dissemination time of agent (i), Δt

is the set time interval and $q(i)$ is quality of decision of agent (i). The duration of the dissemination state is calculated like this:

$$t_{diss}(i) = \Delta t \cdot q(i). \tag{3.1}$$

The agent forms a neighborhood that consists of other agents that have entered the communication radius with state = dissemination. The size of the communication radius determines the size of the area around the agent where the agents inside form the neighborhood. For our experiments, we chose a size of 0.5 meters. It is limited because a global communication radius is not realistic when applied in real-world applications. At the end of the dissemination state, the agent uses a DM to form a new estimated decision about the actual ratio of black tiles in the arena, which are then saved in a new list called DecisionList (DL). In the beginning, each agent votes for a random hypothesis to ensure a diverse distribution from the start. The algorithm stops once the swarm reaches a consensus on its current decision. Thus, when all entries of DL are the same, or when the maximum amount of iterations $T_{max} = 60.000$ has been reached, it stops.

The algorithms, including the arena, the agents, and the DMs are then implemented in Python. We used the popular library NumPy because it provides efficient implementations for mathematics and operations on them.
As agents, we used simulated e-puck robots. Because of the high costs of robots for experimentation, Mondada et al. have introduced the e-puck in [13]. The e-puck is a low-cost robot used for all possible experiments that require a swarm of robots with standardized equipment of sensors and mechanical parts. It is highly versatile because of the possibility to equip it with further needed parts. It can also be used for simulations or physical applications, considering the availability of e-puck simulations by Webots and Enki. These robots are used in many of experiments mentioned in Chapter 2. The e-puck is equipped with the following:

- eight infrared (IR) proximity sensors that are placed all around the robot to measure the infrared light of the environment.

- a 3D accelerometer to track the acceleration.
- three microphones in different places to track sounds.
- a color CMOS camera with a resolution of 640x480 pixels that can be used to experiment with partial vision. However, the usage of it is limited due to the small memory size of the robot.

The e-pucks have a diameter of 75 mm and are moved by two stepper motors, which have a resolution of 1000 steps per wheel. It can also be equipped with many individual extensions as needed, but for our experiments, the basic version should suffice.

Our e-pucks used the same low-level control mechanics as the robots that Valentini et al. [28] used in their experiment. The movement consists of walking in a straight line and rotating on the spot to a random direction, with both directions having an equal probability of being chosen. The duration of the movement is also chosen randomly. Right after the robot finishes turning, it continues walking.
Ebert et al. [9], Shan et al. [25], Bartashevich et al. [2] and others used 20 e-pucks for their tests as we do as well. We decided to use the same number for our experiments. Bonarini et al. [4] mentioned that the amount of robots used in the experiment represents the swarm size and that larger sizes of the swarm are desirable because it increases the level of redundancy and robustness. Valentini et al. used 100 robots in order to experiment with their approach to solve the best-of-n decision problem in [29].

3.1 Test Parameters and Determination of Samples

In order to obtain useful data for analyzing the tested DM, we sampled different combinations of parameters several times. We changed the parameters for the samples to achieve meaningful data that represented the swarms' DM abilities. The tested parameters included the following:

Black-tile-ratio in the arena.
In our experiments, the white tiles always form the majority of the two colors in the arena. We increased the proportion of the black tiles step-by-step, with each step being

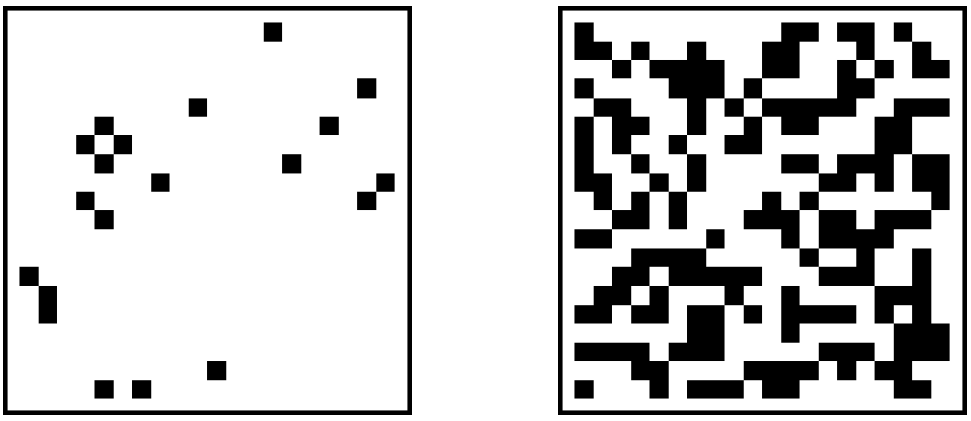

Figure 3.1: Arena generated with a random pattern and a BTR of 0.05 on the left and with 0.45 on the right.

0.10 from 0.05 black/0.95 white to 0.45 black/0.55 white. The effect of BTR changes for the environment is shown in Fig.3.1. When sampling this environmental change, we measured how these differences influence the performance of the DMs. For every sample, the environment was regenerated with the corresponding settings. An increasing share of black and white tiles can also be described as noise. The more similar the number of different colored tiles, the noisier the environment.

Different time intervals for the exploration and dissemination states.
When increasing the length of both states, we changed the time the agents use for enhancing the quality that the agent forms by itself and the time that it uses to form a new opinion when communicating with the neighborhood. We raised the state-interval-times from 1 to 10 seconds. In our experiments, one step was 0.01 seconds long. If we changed the duration of the states, it meant two things: The agent has more time to roam the arena and to strengthen its own quality estimations for the hypotheses during exploration. With a longer dissemination time, it has more time to search for other agents, and it can increase the size of its neighborhood, which increases the number of participants for the DM. The final time an agent spends in dissemination state is ultimately influenced by the quality of the current decision it is adverting. At the start, the time intervals are unbiased.

Environment patterns formed by the tiles of different colors in the environment.

As seen in the work of Bartashevich et al. [2], the difficulty of the collective DM changes with the patterns formed in the arena by the black tiles (respective to the white tiles). Therefore, we decided to use the random pattern and the block pattern for benchmarking our DMs Picture: Random and Block. When looking at the difficulty of the patterns in terms of reaching a correct collective decision, a block pattern formed by the environment is far more difficult for the swarm than a random pattern. Given these facts, we decided to save computation time when benchmarking our DMs and test only the exploration and dissemination time interval with a single time interval. We made 20 runs for each of the setting combinations. An exception was made for the independent decision-making (INDI), which does not use the different states and instead only has exploration and quality estimations for each agent. Because of the abundance of the dissemination state, INDI has no use for the time intervals either. For that reason, INDI has only been tested with different BTRs and MBWs for the random pattern. Some examples for an arena with a block pattern can be seen in Fig.3.2 and Fig.3.3.

Different mean-block-widths for the block pattern.

When benchmarking our implemented collective DM strategies on the block pattern, we changed the actual sizes of the blocks formed by the tiles which can be seen in Fig.3.2 and Fig.3.3. These adjustments and the higher number of experiments increases the diversity of the generated information for the performance of the block pattern. The different mean-block-widths we chose to test our algorithms are 1, 3, 6, 9, and 12. It must be noted that a mean-block-width of 1 is the same as a random pattern because the blocks do not differ from the original size. Because of some restrictions regarding the algorithm forming the blocks for lower BTRs, we decided that we would only test the block widths for BTRs of 0.35 and 0.45. Increased diversity in our environment improved the expressiveness of the benchmarks of our collective DM strategies. The better the algorithms performed in different environments, the higher the estimated quality of the outcome was when using the strategy for a real problem. We also used a fixed set time interval during the two states for all tested algorithms. In order to find the time interval which will be used in the experiments with the block pattern we normalize the results from the random pattern and choose the best option.

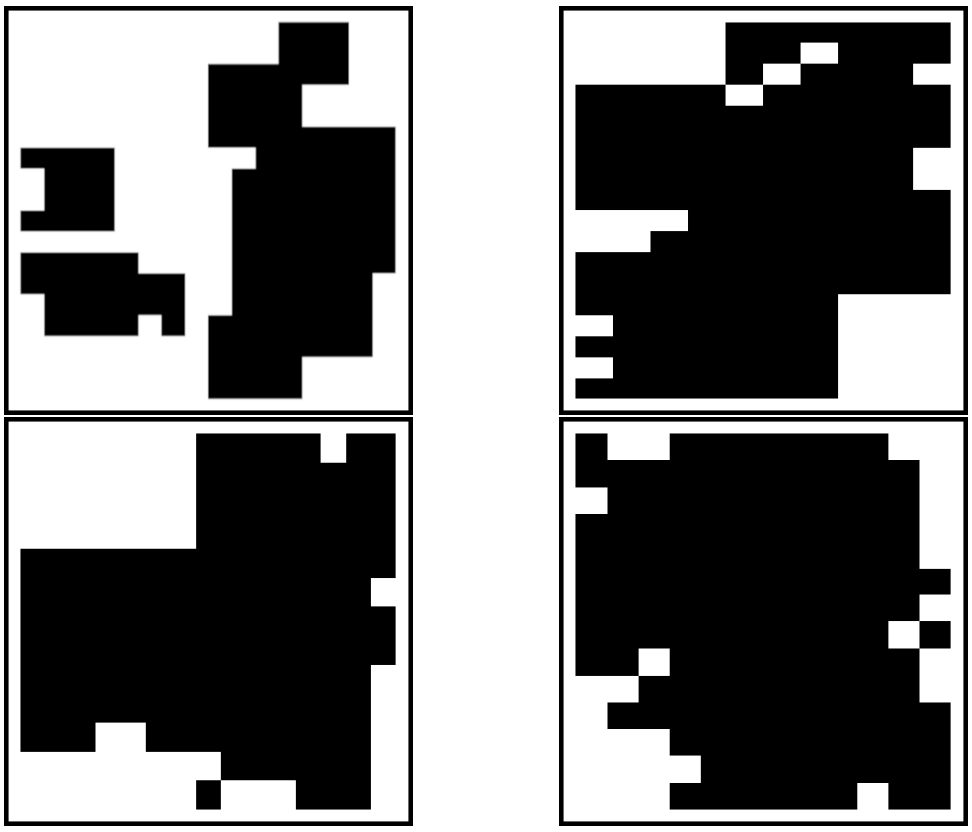

Figure 3.2: Arena generated with a block pattern, a BTR of 0.35 and from up left to down right: a MBW of 3, 6, 9 and 12.

3.2 Decision-making Metrics

Several different metrics have been used to analyze DM strategies and their quality. One of these metrics is convergence time. It tells how fast the algorithm is in reaching a convergence with the swarm opinion considering the best-estimated decision. The lower the convergence time, the faster the algorithm is in finding a consensus. Valentini et al. [30][29], Shan et al. [25], Parker et al. [15], and Scheidler et al. [21] have used this metric to measure the quality of their proposed strategy. Bartashevich et al. [2] also used it to measure the effectiveness of the proposed benchmark patterns. Another way to benchmark a DM algorithm is the exit probability used by Shan et al. [25], Bartashevich et al. [2], and Strobel et al. [26]. It is used to measure the ratio of runs when the swarm comes up with the right decision. The experiments in [25] and [2] used it so that if the swarm chooses the right decision, it will count as successful. In these cases, there were

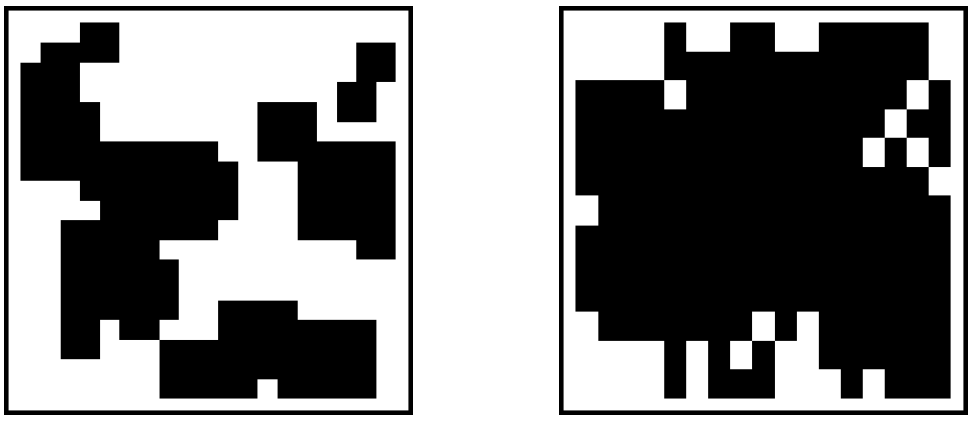

Figure 3.3: Arena generated with a block pattern, a BTR of 0.35 and from left to right: a MBW of 3 and 12.

only binary decision choices: the arena has more white than black tiles or less. It was measured for different BTRs in both experiments. As proposed by Valentini et al. [29] and mentioned by Shan et al. [25], fast swarms are usually not accurate and vice versa. Valentini et al. [29] measured a trade-off for the k-unanimity rule and the influence of the initial distribution of the opinions across the robots. They also investigated the impact of the spatial density of robots on speed and accuracy. In contemplation of the amount of the extracted data delivered by our experiments and the actual benchmarking of the DMs, we needed to establish meaningful metrics. The requirements for these metrics have to be generally applicable to as many DMs as possible, understandable, and comparable. Considering these properties, the nature of the DM strategies and the data we extracted we have chosen the following metrics:

Reliability rate (RR):

It analyzes whether the DMs perform consistently well. In order to calculate RR, we compared the runs where the swarm converged to the runs where the swarm did not converge. Accordingly, S was the number of runs where the algorithm exits because it found a consensus and D the number of all runs. Thus, the metrics RR can be defined as follows:

$$RR = \frac{S}{D}. \tag{3.2}$$

RR represents the percentage of how many runs in total were successful. The higher the RR, the more reliable the DM for the given parameter configuration.

Accuracy (AE):

It measures the mean absolute error of the DM. Our accuracy rate is determined through the calculation of the absolute error AE. If the error is small then the accuracy is high. While our accuracy rate has no accountable value we use the calculated AE to determine the accuracy.

In order to calculate AE for one run we subtracted the final decision FD from the actual correct decision CD in the environment. The absolute value of the outcome is the absolute error AE. Subsequently, we calculated the average of all previously generated absolute errors. The generated mean absolute error ($\overline{AE}$) is our value for accuracy for a set of runs. We defined K as the number of absolute errors. We did not consider the failed runs for the calculation of these metrics, and for this reason, the metrics AE and $\overline{AE}$ can be defined as follows:

$$AE = |CD - FD| \tag{3.3}$$

$$\overline{AE} = \frac{\sum_{n=0}^{N} AE_n}{K}. \tag{3.4}$$

We calculated the accuracy that way because we had a discrete distribution of hypotheses that were all equidistant and represented by their index in H. We use the absolute value of the outcome to enhance the comparability of the results. The value of $\overline{AE}$ describes how accurate the converged runs were that we observed in this test. While the reliability measures if it converges the accuracy determines if the swarm converges to the correct result.

Convergence time (CT):

It measures the mean time in which a converged swarm reaches the consensus. We define the time which the swarm needs to converge in a single run as CT. The mean convergence time of all relevant runs is defined as $\{CT$. N will be the number of all converged swarms which are relevant for $\overline{CT}$. The $\overline{CT}$ is calculated as follows:

$$\overline{CT} = \frac{(\sum_{n=0}^{N}(CT_n))}{N}. \tag{3.5}$$

In this way, we generated a value that was comparable to all other DMs, which were tested under the same circumstances; thus, we could also compare it with experiments that are larger and have more samples to justify the accuracy of the data. It is important to mention that we did not measure the actual speed of the algorithm in real time. We measured the speed in time-steps. Real time is easily influenced by the algorithm's efficiency, the overall capacity and the specification of the machine it is running on. These factors can significantly impact the results. Using speed as a metric for benchmarking our DMs enables its comparability to other DMs, which were not tested in this paper. Furthermore, the time a swarm needs to converge is an important property that can decide about the usage of the DM at all. With these three metrics, as well as the set of the data we obtained from our experiments, we were able to analyze the relevant majority voting algorithms in the same way, which makes our benchmarks precise, simple, and comparable.

3.3 Majority Voting Algorithms

In the section we introduce the three implemented majority voting algorithms for the collective decision-making of the swarm. They are based on different systems of democratic voting. We analyze the functionality of the three decision-making strategies and explain shortly the two we use for comparison.

3.3.1 First-past-the-post Voting

Our algorithm inspired by the first-past-the-post voting is the most straight forward majority voting approach. For this decision-making, the agent creates a list which includes all recent decisions of the participants in its neighborhood including its own decision. After that it chooses the decision which is most represented in the list as its new decision $D(i)$ of i where i is the agent in the swarm. The first-past-the-post voting depends on the influence of the adverted decisions quality. If it adverts a good decision with a high quality the agents dissemination time is longer and it shows up in more neighborhoods.

Algorithm 1: First-past-the-post voting

```
Data: List with possible hypotheses H
Result: decisionOfAgent(i) = New decision for agent i (D(i))
timeInterval    {1,2,3,...,10}
begin
    for each agent i do
        if explorationState == true then
            startTimer (timeInterval)
            agent DO random walk
            agent DO observe the environment
            agent DO build quality for hypotheses
            if timer = 0 then
                explorationState = false
            end
        else
            startTimer (timeInterval * qualityOfDecisionOfAgent(i))
            neighborList = agents with distance ≤ communication radius and in
              dissemination state
            decisionOfAgent(i) = decision with the highest occurrence in neighborList
            if timer == 0 then
                explorationState = true
            end
        end
    end
end
```

3.3.2 Single Transferable Vote

The majority decision-making algorithm STV is the most complex of the three implemented algorithms. If using STV for a democratic election, the voting system handles the election for a number of different candidates, who then form the government. It selects a fixed number of the most appealing candidates out of all the representatives standing for election. The ballots each voter receives provide them the opportunity to rank the candidates instead of electing one possible candidate. When applied in the real world, the voters typically have three ranked votes.

During the election, it takes several iterations of distributing the votes. At the start of

Algorithm 2: Single transferable vote

```
Data: List with possible hypotheses H
Result: decisionOfAgent(i) = New Decision for agent i
timeInterval   {1,2,3,...,10}
begin
    for each agent i do
        if explorationState == true then
            startTimer (timeInterval)
            agent DO Random Walk
            agent DO observe the environment
            agent DO build quality for hypotheses
            agent DO build agentMemory with hypotheses ranked after quality
            if timer == 0 then
                explorationState = false
            end
        else
            startTimer (timeInterval * qualityOfDecisionOfAgent(i))
            neighborList = agents with distance ≤ communication radius and in
             dissemination state
            votingMatrix (i) = 0
            for each possible decision do
                for each neighbor do
                    SUM votingMatrix(decision) and neighborMemory[j](decision)
                    (j)   votingMatrixdosum
                    if decisionScore ≥ 66% ∑ all decisions then
                        decisionOfAgent(i) = votingMatrix(decision)
                    else
                        append votingMatrix(decision) to badDecisions
                        votingMatrix(i) = 0
                    end
                end
            end
        end
        if timer == 0 then
            explorationState = true
        end
    end
end
```

each iteration, it checks if any of the candidates reach a given threshold. In this scenario, the candidate is part of the newly elected government; if not, the count continues. Afterward, the worst candidate is determined and removed from the election. If voters vote for a removed hypothesis, then in the next iteration, the second choice of this agent is used as its new decision. In a real-world application, this process is repeated until all seats are taken. If some seats are empty at the end of the vote-counting, but none of the remaining candidates reaches the quota, the worst one is eliminated, and the remaining candidate (or candidates) wins the last round of the voting system even if it would not have enough votes regularly. We implemented the voting system through various uses of matrices and lists.

In the exploration stage, we created a matrix where each row represented the memory for each of the agents and another matrix that did the same for the estimated quality. Next, we sorted the decisions in the decision memory according to their quality. After the sorting, the first entry holds the decision with the highest estimated quality and so on. This can be expressed as follows. We define m_t as decision-memory, qm_t as quality-memory and M_t as the decision-memory-matrix, each for timestep t. Further, we consider $h \quad H$ a hypothesis of a fixed-ordered finite set of hypotheses H and j the index of h in H. Let $|H|$ be the number of hypothesis in H and q_j the quality of hypothesis h.

With

$$m_t = \{1, \ldots, j, \ldots, |H|\} \tag{3.6}$$

$$qm_t = \{q_1, \ldots, q_j, \ldots, q_{|H|}\}, \tag{3.7}$$

we now consider the sequence

$$\bar{m}_t = \{\ldots, j, k, \ldots\} \tag{3.8}$$

to be sorted so that $q_j > q_k$ and, therefore,

$$M_t = \begin{bmatrix} \bar{m}_{1t} \\ \vdots \\ \bar{m}_{it} \\ \vdots \\ \bar{m}_{Nt} \end{bmatrix} \quad N^{N \times |H|} \tag{3.9}$$

with $\bar{m}_{it}$ being $\bar{m}_t$ for the agent $i \quad 1, \ldots, N$.

We then excluded all agents from this neighborhood that were not found during the dissemination state. We used a new matrix that holds the neighborhood memory and a list where we conducted the voting. Afterward, we overviewed the first entries of the neighbor's memory, which holds the favorites of the agents. We increment the entry of the list by one for each time the decision for which we count the number at this time is encountered in the memory. If one decision reaches a threshold of 66% of the votes, it is the new winner and replaces the decision of the agent. If not, then the decision with the fewest votes is eliminated. We achieved that by saving the bad decisions in a separate list. In preparation for the new turn, we set the voting list back to zero because we had a new turn. If a neighbor now wants to vote for a decision that was a bad decision in one of the past turns, it votes for the next decision, which is not part of the bad decisions. This continues until a decision reaches the threshold or until all decisions have been checked. If no winner is found, the agent stays with the old one.

3.3.3 Borda Count

The implemented collective DM approach designed after the BC voting method uses a score and ranking system to select a winner among the possible frequencies of black tiles in the environment. In a real election, this voting system allows each of the voters to rank the possible candidates by preference on their ballots. After collecting all ballots, the

Algorithm 3: Borda count voting

```
Data: List with possible hypotheses
Result: decisionOfAgent(i) = New Decision for agent i
timeInterval    {1,2,3,...,10}
begin
    for each agent i do
        if explorationState == true then
            startTimer (timeInterval)
            agent DO Random Walk
            agent DO observe the environment
            agent DO build quality for hypotheses
            agent DO build agentMemory with hypotheses ranked after quality
            if timer = 0 then explorationState = false
        else
            startTimer (timeInterval * qualityOfDecisionOfAgent(i))
            neighborList <- agentswithdistance ≤
             communication radius and in dissemination state
            bordaCount(i) <-0
            for each possible decision do
                for each neighbor do
                    aggregate bordaCount(decision) and weighted neighbor(decision)
                end
            end
            decisionArray(i) <-decision with smallest sum of weights in bordaCount
        end
        if timer == 0 then
            explorationState = true
        end
    end
end
```

election commissioners distribute points based on the rank the corresponding candidate achieves on the ballot. In the final step, the points of all ballots are summed up. The winner is the candidate with the highest score. Our algorithm worked in a similar manner. First, in the exploration state, we created a memory matrix in the same way as we did for the STV algorithm. During dissemination, the agent still wanders around and tries to build a neighborhood out of suitable agents. These agents will be taken of the memory to build a new matrix representing the neighborhood's memory. We then created a list of zeros, where each entry represents the decision respective to their index. This works because our agents vote for an entry in the list of possible hypotheses. If they vote in our experiment for 0.05, the vote for the decision is 0. We went through each of the rows in the memory of the neighbors and incremented the score in the list where the now checked decision stands by the index of the read entry. This means that if there is decision 4 in second place, it receives one point because the index of decision 4 is one. Afterward, we checked for the decision with the lowest score and found our winner. If two had the same score, we used the first one found by the method. The old decision of the agent was then replaced by the winner of the BC voting.

3.3.4 Direct Modulation of Voter-based Decisions

The DMVD (direct modulation of voter-based decisions) approach was created by Valentini et al. [30]. It is quite similar to our FPTP strategy. The only difference is that instead of choosing the decision with the highest occurrence in the neighborhood, the agent chooses a random decision among the neighbors. The method also depends on the quality of the adverted decision. The higher the quality, the longer the dissemination state is.

3.3.5 Independent Decision-making

The INDI (independent decision-making) is our second voting algorithm, which we compared with our proposed strategies. Contrary to the most collective DM algorithm we mentioned until now, INDI does not rely on communication between the agents. It also does not have a dissemination state. The agents still wander over the arena and observe

the environment to estimate the qualities for each possible hypothesis. The decision of the agent is the decision with the agents' highest estimated quality.

4 Analysis and Results

During this chapter, we present and analyze the different results created while sampling the majority voting algorithms with the different parameters as explained earlier. We use reliability, accuracy, and convergence time (CT) as metrics to benchmark them.

4.1 Determination of the Quality for the Majority Voting Algorithm in a Random Environment

Because of the similarities of our FPTP approach and DMVD, we expect similar results for both. However, because the FPTP algorithm relies on the majority of the decisions in the neighborhood, it is not only dependent on the duration of the time interval (TI), but it also has a direct impact on the DM process; therefore, we expected a small CT. The accuracy will not be improved significantly compared to DMVD because FPTP relies more on the initial distribution of decisions among the agents, which strengthens the chance of finding a wrong consensus early. This behavior also decreases the estimated accuracy of this majority DM approach. In terms of reliability, we expected it to show good results and perform similar to DMVD.

In contrast to most of the other algorithms like DMVD or the k-unanimity rule, which do not use the quality of the hypotheses during dissemination, our two implemented strategies inspired by STV and BC are heavily reliant on it, just like the independent DM.

These algorithms require the agents to rank the hypotheses and perform a complex procedure to determine the new decision among the given decisions in the environment. The ranking works the same way as the independent DM. Agents need some start-up time to observe the environment in order to rank the hypotheses. Our results are expected to show that this process negatively impacts the CT for both strategies that use the ranking.

In regard to the functionality of STV, the outcome of each DM depends on the size of the neighborhood and the estimated quality of the hypotheses. If the neighbors have a similar quality estimation, the DM ends quickly with a decision that reaches the threshold. If they have a completely different quality estimation of the hypotheses, it is unlikely that a threshold is reached, and the old decision is kept. The DM changes with the number of agents in the neighborhood because the threshold was set at 60%. If the agent is alone, the new decision will be the favored hypothesis. If there are two, then the voting party needs a consensus to actually reach the threshold. When the neighborhood grows, the algorithm works better. However, most of the time, agents need to work with small neighborhood, which makes reaching a consensus difficult. We concluded that a solution slowly dominates the field. We expected that this leads to a higher consensus time. While the time the swarm spends to find the globally optimal decision is estimated to be high, the ranking and the implemented threshold reduce the possibility of finding a wrong consent. In sum, we estimated that the STV algorithm performs accurately and reliably but has a slower CT.

The BC has a high reliance on the qualities of the hypotheses because it uses the same ranking method as STV but does not depend on the size of the neighborhood. In our algorithm, which was designed after the BC majority voting system, each agent performs the DM no matter how many other agents participate in the voting. If an agent is alone in the neighborhood, its new decision becomes the favored hypotheses with the highest estimated quality, just like in STV. In contrast to STV, the scoring system works efficiently and precisely, even with two participants. Even after an election where the last candidate of an agent wins the race, it will not change the estimated quality of the hypotheses for each candidate. The favorite and the ranking stays unchanged. If the estimations of the other participants of the neighborhood differ much from the correct hypotheses, the

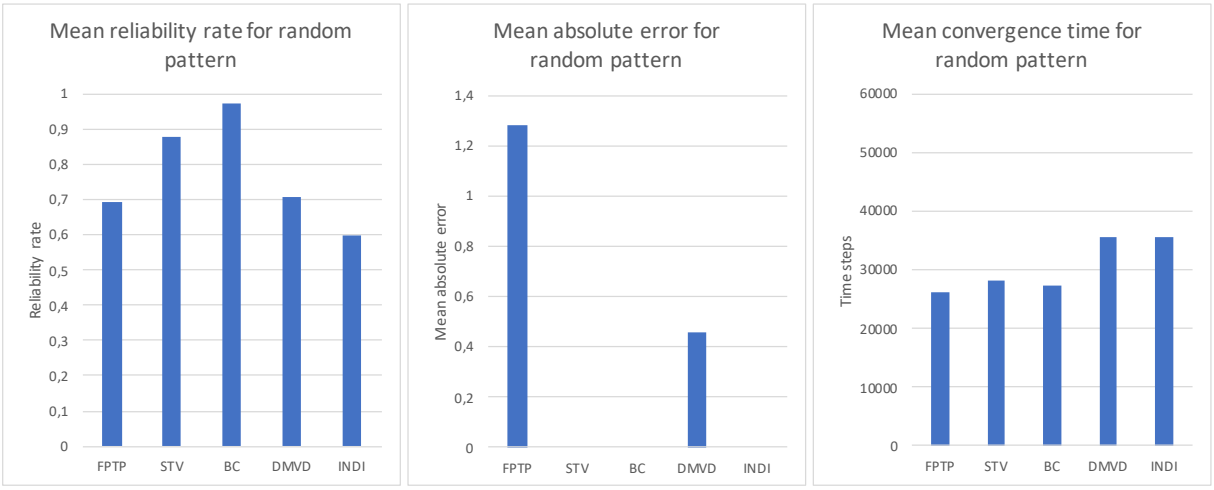

Figure 4.1: Mean reliability rate, mean absolute error and mean convergence time for all 5 analyzed algorithms based on combined samples for the random pattern. The values result from 1000 runs per algorithm with different black-tile-ratio and time intervals for the first three algorithms. INDIs values result from 100 runs.

chances are high that the agents will have another vote at a later time where neighbors with better rankings participate. That way, a hypothesis with a good mean quality will dominate the agents' decisions over time. We estimated that the BC would not be quick when it comes to CT but quicker than the STV. The lower CT is expected to correlate with higher reliability.

The agent will return to its favorite hypothesis every time when it is alone in the neighborhood, even if its estimated qualities are far apart from the values, which would lead to a correct decision. It should be mentioned that the quality of the hypotheses changes as well as the favorite hypotheses of the agents. In this case, the agents' decision will be corrected when more agents participate in a later vote. The scoring system has little use for the favorite hypotheses of the agents. Instead, it aims to find the best decision upon which all agents can agree upon. This mechanic is attuned to poor qualities and makes a widely accepted bad decision among the agents unlikely, which leads to an expectancy of high accuracy for the BC algorithm.

Looking at the experiments regarding the reliability of the FPTP algorithm, it is nearly equal to DMVD and better than the INDI algorithm. The measured average reliability of all our samples of FPTP was 0.694. Considering that both are quite similar, this result

met our expectations.
In terms of accuracy, which only includes the cases where the swarm found a consensus, it was by far the worst of our analyzed strategies. With a $\overline{AE}$ of 0.906, it was less accurate than DMVD with 0.332. This is because the agents in our FPTP strategy tend to be quickly dominated by a decision. If agents create a majority through voting, this can expand quickly over all other agents because the others will join the majority. Influences on the modified time interval become apparent later, so that the starting phase tends to be the most critical for the FPTP algorithm. These measurements are associated with a trade-off for the convergence speed.
Considering the samples with the correct consensus found in time, the FPTP was the fastest of our tested strategies. With an $\overline{CT}$ of 26,216.706 steps across all samples, it was 9,418.144 steps faster than DMVD, which had an $\overline{CT}$ of 35,634.85. The results we see in Fig.4.1 shows how quickly a decision can dominate the swarm if the agents only follow the majority compared to a random selection.

The analysis of the data for the STV generated by our experiments shows that it is a high-quality DM strategy compared to FPTP, DMVD, and INDI. The algorithm has a high reliability rate of 0.877, which fits our expected results. The threshold limits the capability of unpopular hypotheses to dominate the swarm and spread quickly. That leads to a robust and reliable voting algorithm, which was confirmed by our results.
While the reliability is high, the accuracy of the STV in a random environment with different BTRs was perfect. We had a $\overline{AE}$ of 0, which means all our converged samples found the correct BTR.
The fact that the accuracy and the reliability are both high is especially interesting if we take a look at the $\overline{CT}$ of 28,104.193 steps. It was only 6,71% slower than FPTP, which was the fastest algorithm and faster than DMVD and INDI. That means we did not have a trade-off and our implemented STV algorithm shows a high quality when used in environments similar to our arena with a random pattern.

Considering the similarities of the exploration behavior and the ranking system, it is relatable that the BC-inspired algorithm performs just as well. The results shown in Fig.4.1 confirm our assumptions taken earlier. With a reliability rate of 0.975, it has the

highest convergence rate of all of our implemented algorithms. This means that only a small fracture of the runs did not converge. It has a small advantage of 0.098 compared to STV. This advantage comes from the scoring system. As mentioned earlier, it allows the agents to keep the focus on the best decision for all agents. This way, the newly voted decision has at least a mediocre quality. Moreover, if the participants are predominantly wrong, then it will be corrected in later voting.
In terms of CT, the BC shows excellent results. A $\overline{CT}$ of 27,290.538 steps shows that it is clearly faster than DMVD and INDI. It was also 2,9% faster than STV but 3,9% slower than the FPTP, which was the fastest. This puts it in second place when it comes to speed. We did observe a trade-off between speed, where it comes second, and accuracy, where it is the first. Furthermore, our algorithm had a $\overline{AE}$ of 0, which shows that it is a highly accurate strategy. This analysis is based on data illustrated in Fig.4.1.

4.2 Determination of the Decision-making Qualities across the Black-tile-ratios used for Random Patterns

The first set of our experiments analyzed the general quality of our proposed algorithm when used in environments with a random pattern. Therefore, we split our data so that we could observe the results and determine the quality for each BTR setting in relation to the others. We performed this for each of the three algorithms and added DMVD and INDI for comparison purposes.

With an increasing BTR between 0.05 and 0.45 for our environment, we estimated that the more difficulties the algorithms had determining the correct BTR, the more similar the share between black and white tiles would be. We expected that result because if an agent passes several tiles of the same color, the quality for a majority of said tiles grows. When it only passes white tiles, then its quality for a hypothesis like BTR of 0.05 grows more rapidly. This is the case if there actually is a high majority of white or black tiles. With this hypothesis, we also followed Bartashevich et al. [2], who showed that

the exit probability, a metric that measures the ratio of correct converged runs among all samples, decreases with the increased density of the invading (here black) color. While they did not employ the same setup and used different metrics, we believe that the exit probability and the reliability rate are somewhat similar in expression. Additionally, their algorithm checked which color was in the majority, and our algorithms tried to find the hypotheses that are closest to the actual BTR. However, we still expected similar results when analyzing our data.

When looking at the impact of the mechanics of the DM we used in our algorithms, we expected small differences if confronted with an increasing BTR. In an environment with a small noise (that is, a small number of different colored tiles), the time interval of the FPTP will increase. As mentioned before, the quality of a hypothesis which represents a smaller BTR is expected to increase faster in such an environment. The correct decision will spread more rapidly and quickly overtakes the swarms' decisions. Because this is the only change in the behavior, we expected that the overall quality would decrease.
The only thing that changes for the algorithms is the estimated quality of the hypotheses. Thus, we expected that the impact of the increasing share between two colors has a similar effect on our STV and BC algorithms because they used the same exploration and ranking method.
The ranking, however, will be influenced by the quality of the hypotheses. We expect that the quality of the correct hypotheses and those which are close to it will increase more rapidly for a low BTR than it would for an environment with a similar share of white and black tiles. It is more difficult to estimate the correct likeliness for a hypothesis. That means the agents find a favorite that dominates the others in the ranking more quickly and that this favorite is mostly the same for each agent. That speeds up the CT and increases the accuracy and reliability for smaller frequencies of black tiles for STV and BC. The implemented ranking was expected to make both algorithms more robust to poor qualities. This behavior will result in a less intense loss of effectiveness compared to FPTP, DMVD, and INDI.
Following these conclusions, we expect a decrease in reliability, accuracy, and convergence speed for all three of our algorithms when the BTR increases with the difference that STV and BC are more robust to the environment changes than FPTP, DMVD, and

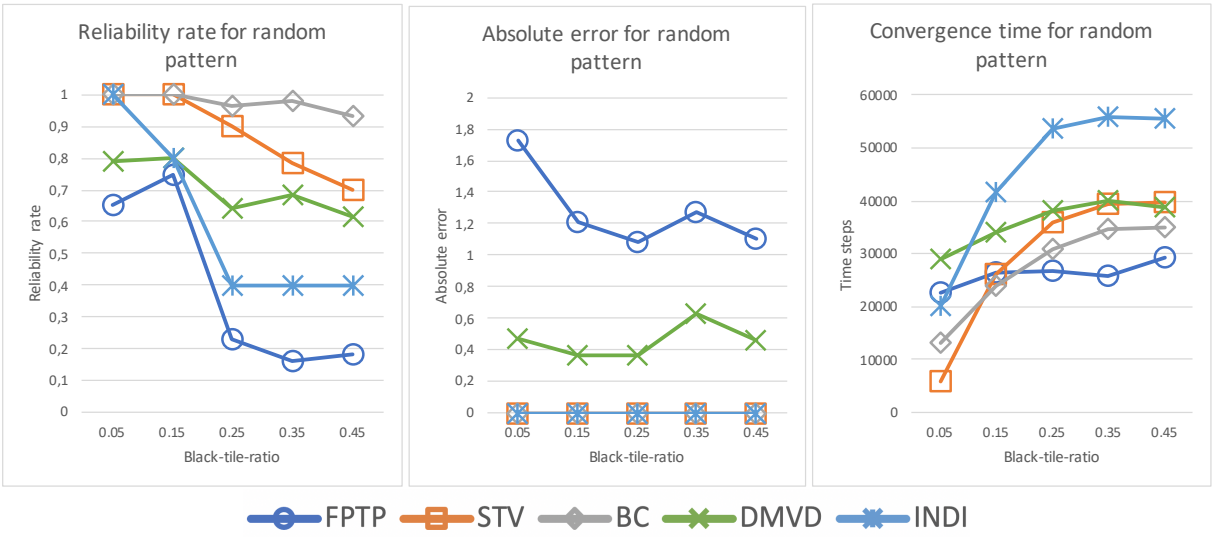

Figure 4.2: Reliability rate, absolute error and convergence time for all 5 analyzed algorithms at different black-tile-ratios for the random pattern. The values result from 200 runs per black-tile-ratio. INDIs values result from 20 runs each.

INDI.

In sum, the results generated by our experiments met most of our expectations in terms of effect of the BTR. Regarding the reliability of FPTP, it indeed decreased with increasing equality of the two sets of colored tiles, just as we expected. It shows a negative trend disrupted by positive spikes. We measured a reliability rating (RR) of 0.65 for a BTR of 0.05 and a RR of 0.185 at 0.45, which is a decrease of 0.465. We then observed some positive changes at 0.15 with an increase of 0.1, which falls again to 0.23. This change contradicts our expectations of a negative trend. The most significant fall of the curve happens between the BTRs of 0.15 and 0.25 with a decrease of 0.52.

The results of the accuracy for FPTP when we changed the BTRs did not show the expected trend. It is a slight downward trend with a fluctuation at 0.35. At this point, the rating increased by 0.09. This increase and the small difference between 0.25 and 0.45 are due to the random components of the algorithm. There is a clear trade-off between accuracy and reliability for FPTP. In other words, when the reliability decreases, the accuracy increases.

In terms of the CT, we also observed a slight upward trend, which ultimately means a negative change. There were small fluctuations at a BTR of 0.35, and we recorded an increase, which starts at a BTR of 0.05 with 22,742.538 and ends at 29,285.672 for the

BTR of 0.45. FPTP shows the best CT out of all algorithms for the BTRs of 0.25, 0.35, and 0.45.

FPTP shows a significantly decreasing trend with a positive spike and some fluctuations at the end. This trend is as expected. The unrelated fluctuation is caused by the random nature that underlies all of our algorithms. The movement patterns are random and can cause the robots to spread far across the arena, or they can cluster in the same area. In FPTP, decisions can quickly dominate the agents when they cluster. This increases the chance for convergence and can lead to a higher reliability rate even if these changes oppose the trend. When the agents observe many tiles of the same color, as is the case when there is a small BTR, the quality of these hypotheses grows rapidly.
The trade-off between the reliability rate and accuracy shows that with a low BTR, the algorithm manages to converge often to a single decision, but these decisions tend to be rather far away from the correct solution. With higher equality of black and white tiles, the reliability rate becomes worse, and the accuracy increases. The swarm is less accurate because the swarm converges quickly, caused by the fast domination of wrong decisions when the qualities increase fast. The average of the decisions of FPTP is generally closer to the middle of the list of hypotheses which is most far away from the extremes (in our experiments this is the fourth and fifth hypotheses) than to the outer hypotheses because the accuracy of the implemented strategy is very low. This mechanic causes the $\overline{AE}$ for small BTRs and for significantly large BTRs to be higher than in a balanced environment. The CT of FPTP shows the least sensitivity to and dependence on the environment compared to the other algorithms. While we still see a negative trend, it is the one with the slightest slope. DMVD shows a similar curve but is generally slower. The changes that make the trend discernible are too small and unsteady to allow us to determine whether there is a connection between the CT and the tested environment changes. We concluded that the decisions dominate the swarm in about the same amount of time for different environments. The CT depends on the paths of the agents and the neighborhoods but does not change with the quality of the adverted hypothesis.
Our FPTP algorithm shows robustness in terms of CT with environment changes, but the accuracy and reliability are not that robust and show a trade-off. This makes FPTP a reliable but inaccurate strategy in an even area which reverses when the environment has

increasing noise. However, compared to DMVD, STV, and BC, it shows no advantage. The results of the FPTP voting are worse or about the same in every experiment.

In terms of reliability, the values we obtained from our experiments with our STV system show that it matches our expectations of a negative trend. We found a RR of 1 for a BTR of 0.05 and 0.15, which means that every measured run in that environment converged. We also found a nearly linear decrease until it reaches a low with a RR of 0.7 at 0.45. We did not observe fluctuations or discrepancies.
Furthermore, STV exceeded our expectations in terms of the accuracy of the implementation. We observed perfect results without any disturbances.
For the CT, we expected a negative trend, which fits the results generated by our experiments. In an environment with a BTR of 0.05, we measured the lowest CT with 5,981 steps. The slope converged at 0.45 with a CT of 39,802.429 steps. Between 0.05 and 0.15, we observed a positive change of 29,8% but between 0.35 and 0.45 only 0,8%.

The reliability of the STV is robust to an increase of the BTR in the lower volume range, although it shows a negative trend. The reason for that is the dwindling certainty of agents in their hypotheses, which occurs when the colors of the environment become more balanced. The more noise in the arena, the more diverse the agents' rankings are. The influence of the BTR makes an agreement with other votes from other agents less likely and reduces the chance that opinions will converge. Our implemented STV strategy across the tested environment settings with the random pattern is more reliable than the tested FPTP, DMVD, and INDI approaches.
The accuracy is not influenced by the different BTR settings for the environment because the ranking system and the threshold function prevent a wrong consensus. This makes the domination of agents with an uncommon ranking unlikely as they get outvoted.
While the accuracy of STV did not show any trend that would let us suspect a decrease in quality for the environment settings, there was a clear trend for the CT which means that environment noise in a random pattern impacts how fast the swarms find their consensus. The gradient of the curve decreases rapidly as we can see in 4.2. This is because, in an environment where one color strongly dominates, the favorite of the agents is found fast and is elected easily. The chance that most of the agents have the same favorite in such

environments is high. With an increasing BTR, the favorites differ more, which makes their election less likely. This result is underlined by the fact that STV shows the fastest CT at 0.05.
We determined that our STV is sensitive to changes of the BTR which is an equivalent to increasing noise. An exception is shown here by the accuracy, which was perfect. The quality of the voting system is still high, which makes it an acceptable DM alternative for a swarm in a random environment. These determinations were made using the results shown in Fig.4.2.

Our algorithm, which used the BC majority voting system as inspiration, was influenced as expected by the BTR settings. We observed for 0.05 and 0.15 a perfect RR of 1. Afterward, BC shows a negative trend, ending with a rating of 0.93 at 0.45. A fluctuation is identifiable at 0.35, with a slight positive change. It has an overall decrease of 7% from the best to the worst measured value.
Similar to STV, this algorithm also has a perfect accuracy rating with a $\overline{AE}$ of 0 across all BTR settings, which exceeded our expectations.
The CT of the BC algorithm confirmed our estimations. We observed a negative trend with decreasing changes for higher BTRs. It starts at 13,086 for a BTR of 0.05 and ends at 34,998.959 at 0.45. There is a flatter curve as for STV and unsuspected fluctuations are not registered.

Our experiments with BC generated excellent results. The study shows that it is the most reliable of our three algorithms and also superior compared to DMVD and INDI. It rarely fails to find the consensus, and if found, it is always the correct decision. This strategy uses its ranking and its implemented scoring system to find the best solution for all participants. The most significant advantage compared to the other algorithms is that it uses all available decisions from the agents. With that mechanic, it can compensate for bad rankings from single agents. This makes it fairly robust and increases the quality of the DM and resistance to noise. The CT is influenced the most by the environment changes. There is a similar curve as to STV here but with smaller gradients and better values. This result can be explained by the less important rule of the favorites of the agents. In BC, there is one voting step where all decisions are included and dealt with.

It finds the best compromise for each voting round. We did not see any trade-off, only negative trends, which we already expected.

The BC algorithm is the best of our tested algorithms and the most robust to environment noise changes. It is extremely precise and is at 0.15 the fastest algorithm. Additionally, it has the most robust reliability with small changes and excellent results overall. Even compared to the next best strategy, STV, which also uses a ranking system, it shows superiority. Only FPTP shows better robustness in terms of the CT. This makes the implemented BC majority voting algorithm the best decision for using as a DM in a swarm like ours. Looking ahead, all algorithms get more accurate when the time interval is longer.

4.3 Finding the Best Time Interval for Exploration and Dissemination

In this section, we analyzed the results, which were generated by testing our algorithm with several different time intervals. We determined the best time interval for further benchmarks. The chosen time interval is selected to be used in the next experiments with the block pattern.

Together with the time interval, the lengths of the exploration state and dissemination state also change. This mechanic affects two factors: the time agents have at a stretch to wander the neighborhood and build up the qualities, and the time they spend looking for other agents who are suitable members of a neighborhood for majority voting. Additionally, different time intervals also change the time that a single agent can participate in the neighborhoods of other agents and spread its own decision or ranking.
The effect of the quality on the dissemination state changes too. For a theoretical time interval of 2 seconds, a quality of 0.5 for the adverted would mean a final dissemination time of 1 second. The same quality for a time interval of 10 would mean 5 seconds. It changes proportionally.

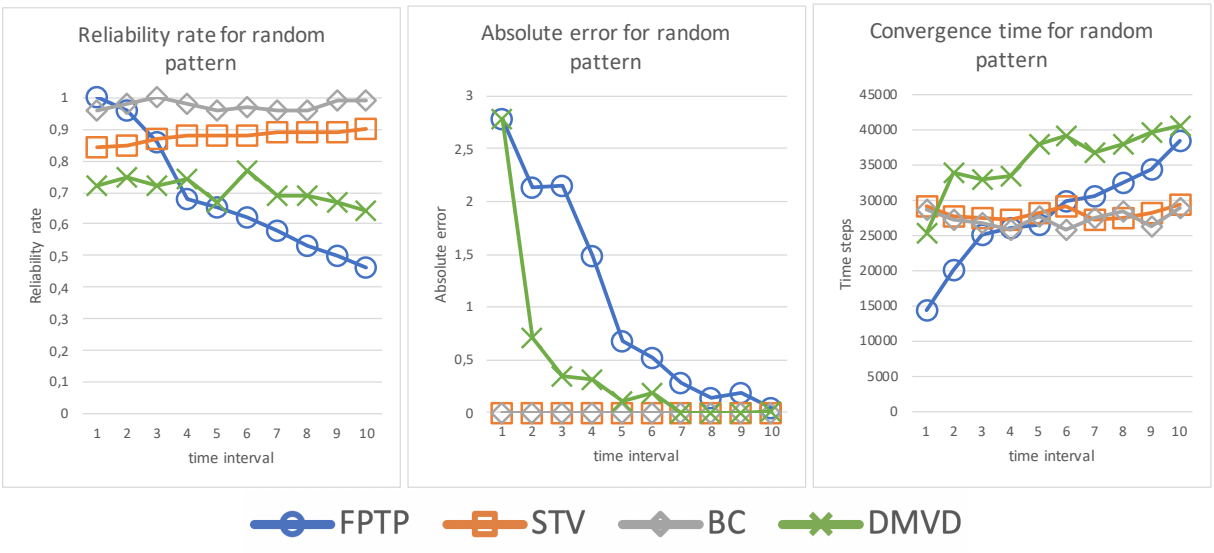

Figure 4.3: Reliability rate, absolute error and convergence time for all 5 analyzed algorithms at different tested time intervals for the random pattern. The values result from 100 runs per time interval.

The quality of the hypothesis j, which we call here $q(j)$, with the set time interval TI has following impact on the dissemination time t_D:

$$t_D = TI \cdot q(j). \tag{4.1}$$

Very short time intervals would make it difficult to build meaningful and accurate qualities. The impact of the biased state times for FPTP and DMVD would be too small to have a useful effect for the decision-making.
This would decrease the possibility of spreading the current decision. If the time interval is short, then the agents' neighborhood is small and its decisions would not appear in enough neighborhoods caused by too few overlaps of the communication radii. On the other hand, a high time interval can bring a wide distribution of bad decisions because the hypotheses, which tend to be worse than the others, will get a relatively longer time in dissemination. We expected to obtain the best results from the upper quarter of the tested time intervals, which should equalize the two extrema. The accuracy of the STV and BC voting will not be affected by the different time intervals because we have not measured any changes before.

The measured reliability rates of algorithms contradicted the expected behaviors. There

is a strong trend at the curve from FPTP and a negative trend with some fluctuation for DMVD as it is shown in Fig.4.3. With a time interval of 1, FPTP has a reliability of 1, which means that each run converges. Afterward, the rate falls to 0.68 at a time interval of 4 and steadily decreases afterward. Opposing to our expectations we found the best results for a time interval of 1 and the worst at 10.
The analysis shows a positive trend for STV. The trend is easily recognizable because it is very steady, although the changes are small. From the lowest reliability rate at 1 with 0.84 and the highest at 10 with 0.9, the increases total 0.06. This curve agrees most closely with our assumptions.
In contrast, BC does not show a trend as FPTP or DMVD, but its course has inconsistencies. The lowest reliability rate of 0.96 is measured at time intervals of 1, 5, 7, and 8. It has the best result at a time interval of 3 with a reliability rate of 1. BC has the most robust values for the reliability rates of all of our tested algorithms, with fluctuations no more than 0.3.

Our FPTP voting algorithm shows a positive trend, which disagrees with our estimations. There is also a trade-off between reliability and accuracy which we can notice when looking at Fig.4.3. When the reliability rises, the accuracy declines, with an exception for the time interval of 3, where we see a fluctuation for the measured AE. Together with DMVD, it records a poor AE, as seen in relation to the others. These results make those two the most inaccurate DM strategies for a time interval of 1. From the TI of 1 to 10, we recorded a decrease of 2.656. It has the best performance, with a time interval of 10, which is an AE of 0.04. DMVD shows a similar trend but performs better overall and has a better performance even for shorter time intervals. STV and BC perform excellent again. The runs have not missed the correct decision once.

As we can see in Fig.4.3, FPTP and DMVD have similar negative trends for their CT. FPTP shows the best CT for the shortest tested time interval with 20,119.75 on average. We recorded increases until it ends with 38,337.95 at a time interval of 10. It has the strongest dependence on the time interval compared to DMVD, STV, and BC.
STV and BC do not show any signs of trends for their convergence time. After analyzing the generated data, it is clear that their CT is fairly robust to changes of the time

interval. The small fluctuations are due to the random nature of the algorithms. Overall, BC performs slightly faster than STV, but the differences are marginal. STV has its best performance at a time interval of 7, with an average of 27,207.82 steps and BC for a time interval of 4 with 28,903.0202 steps.

Unlike the FPTP voting, BC and STV show no discernible trade-offs. From this, we can conclude that FPTP is more dependent on the used time interval than the other two are. The data generated by FPTP can be explained as follows.
Due to faster communication and shorter time intervals between disseminations, agents have little time to form informed opinions when the time interval is low. The effect of the quality on the dissemination time is small. The random decisions of the other agents, if present in the starting distribution such that nearby agents have the same ones, will quickly overrule the others even if they are wrong and have a small quality. Decisions generally tend to dominate the field at the FPTP algorithm under the stated circumstances. This effect is favored by a low time interval and reflected by the CT. On one hand, we have a high percentage of converging runs when the interval time is low; on the other hand, we have one of the highest measured absolute errors. Accuracy and reliability have opposing trends. With growing time intervals, the impact of the quality on the state duration increases which leads to a longer time needed to form quality and finally to converge.
With a high time interval, fewer converge in general, but those that do have a higher chance of finding the correct decision.
The small influence on STV and BC can be explained by the fact that they are not dependent on the time interval to work correctly. While the dissemination time is determined by the quality and the set time interval, the effect on it is small. Only the small positive increase in the reliability of STV shows that the IT has the effect we expected. Through ranking and the use of a threshold, STV balances the possibility of being biased by absent or false robots. Unlike FPTP, the dissemination time for STV and BC is based on the quality of the current decision, but this does not represent the favorite nor what they bring to the voting process. If the current decision resulting from an election during dissemination time ranks poorly for an agent, then the agent will spend more time (proportionally speaking) improving its qualities than if it had its current favorite as the

current decision. This mechanic plays a more important role for the agents as individuals than for the swarm and is strengthened or weakened by the time interval, depending on the duration. The swarms using STV and BC do not benefit much from that mechanic.

The following conclusions can be drawn from the data we present in Fig.4.3. For FPTP, there are two opposing trends, which form a trade-off. If many runs converge, they are mostly wrong; the reverse is true as well. To find accurate results, they also need longer to converge reliably. When choosing the time interval, one must set a preference or use a balanced option, which would be a mediocre time interval.
Because of the same performance differences for STV and BC, we can draw similar to identical conclusions for both of them. We cannot see any strong trends here. Only the STV algorithm shows a correlation between reliability and the time interval, as we expected. This result means that the accuracy and the CT for STV and BC do not depend on the duration of the states.

4.4 Determination of Decision-making Qualities in an Environment with Block Pattern

In this section, we will test our algorithm on the block pattern proposed by Bartachevich et al. [2]. As mentioned in Chapter 3, patterns formed in the environment are expected to have a strong impact on the performance of the algorithms. Using this approach, we were able to benchmark our algorithms in a new environment, thereby improving our ability to determine the usefulness of the majority voting algorithms for the DM in a swarm when used on different problems. To improve the analyzability further, we presented the results using the different mean-block-widths (MBW). It also highlights the effect of the arrangement of black tiles. The larger the mean-block-width, the larger the size of the blocks. In other words, if we have a lower MBW, then we will generate several smaller blocks instead of a large one. With the results we recorded, we chose the time interval of 8, which in our opinion offers the best compromise considering all factors like the bad

performance of FPTP.

The functionality of the tested algorithms does not change when using a block pattern. It only changes the arena. Larger blocks of the same color make it difficult for agents to correctly estimate the quality of the hypotheses. An agent that moves during its exploration state only on one of these blocks will give the corresponding hypotheses a high quality. The larger the blocks, the higher the probability that this happens. Thus, we expected that the performance of all algorithms would be worse compared to the random pattern. Additionally, we expected a correlation between the performance and the MBW. When the blocks are larger, then there is more space for multiple agents to walk on them, and those agents can approve the quality estimations of the agents with similar qualities. The FPTP algorithm has shown a high dependence on the quality of its adverted decision and, thus, on the duration of the dissemination state. Therefore, we expected that the block pattern and the MBW would have the most significant impact on it. Furthermore, the data FPTP generated for the random pattern was rather sensitive, and no competition for the other two.

We estimated that the STV and BC would be more robust to the blocks as they have shown robust results for different BTRs with the random pattern. If two agents with strongly opposing rankings meet, which can happen with an increasing probability in a block pattern and form a neighborhood during dissemination, then both algorithms would find the best decision for both of the agents. They would elect a decision with mediocre qualities rather than the favorites of the agents because they would be the worst decisions for the other one. The elected hypothesis is a compromise for all agents that participated in the election. Of course, this works for limitless agents. However, a larger neighborhood would improve the DM. We expected the MBW to have a smaller impact on STV and BC than it has on FPTP and DMVD. These estimations include reliability, accuracy, and CT. Furthermore, we expected that the influence of the pattern and the MBW would not affect the reliability of FPTP as much as it does for STV and BC. Previous results have shown that solutions quickly dominate agents in FPTP. However, at a time interval of 8, we have lower reliability but increased accuracy for FPTP compared to 1–7. We expected more reliable but less accurate results.

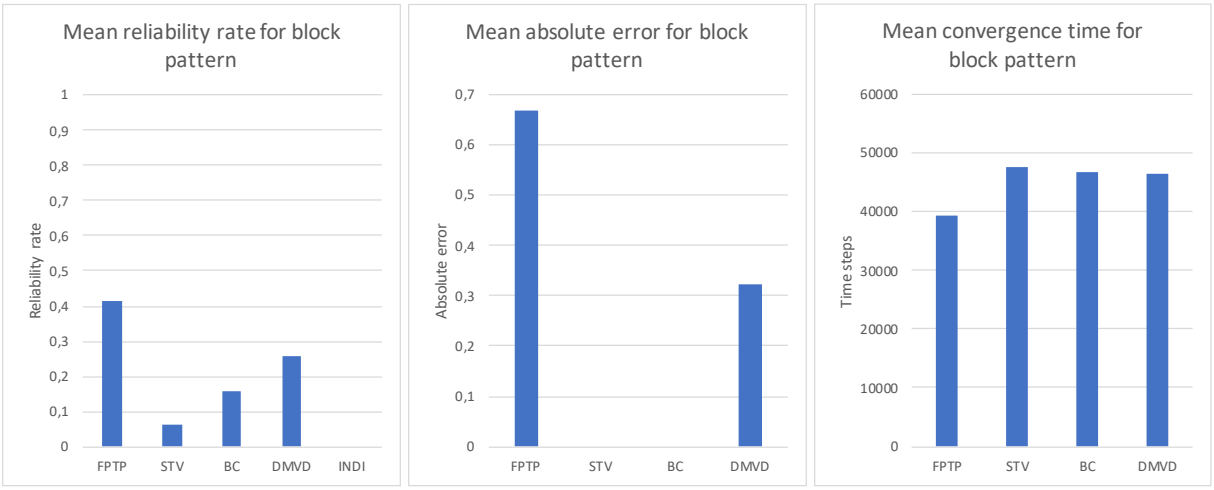

Figure 4.4: Mean reliability rate, mean absolute error and mean convergence time for 4 algorithms based on combined samples for the block pattern. The values result from different black-tile-ratios and mean-block-widths. The INDI approach is not pictured because it didn't converge once for the block pattern. The values result from 100 runs per time interval.

With the experiments we conducted and the results we generated which are shown in Fig.4.4, we can confirm that the pattern and the MBW have a significant impact on our algorithms. At this point, it should be mentioned again that an MBW of 0 is a random distribution of the black squares and is used here only for comparison purposes. Looking at the overall performance of each algorithm in an arena containing a block pattern with a time interval of 8, we noted the following:

In the data we collected for reliability, we found that the independent algorithm added for comparison purposes performed the worst. The swarm did not succeed even in one attempt to find such matching qualities that could converge. Because INDI represents the pure quality analysis performed by the agents without communication, we can take it as the base for the other algorithms. The most reliable strategy is FPTP, with a mean reliability rate of 0.412. DMVD takes second place with 0.256, followed by BC with a rate of 0.156. The worst performance and the heaviest impact on the performance shows the STV algorithm with a mean reliability rate of 0.0625. Only 10 of 160 runs managed to converge. This result was unexpected and did not correlate with the results of the random pattern. In terms of accuracy, STV and BC dominate the others. We see a $\overline{AE}$ of 0, which shows that their superior accuracy is resistant to the block pattern. With a $\overline{AE}$ of 1,279, FPTP is quite inaccurate, even compared to DMVD with 0.459. Similar

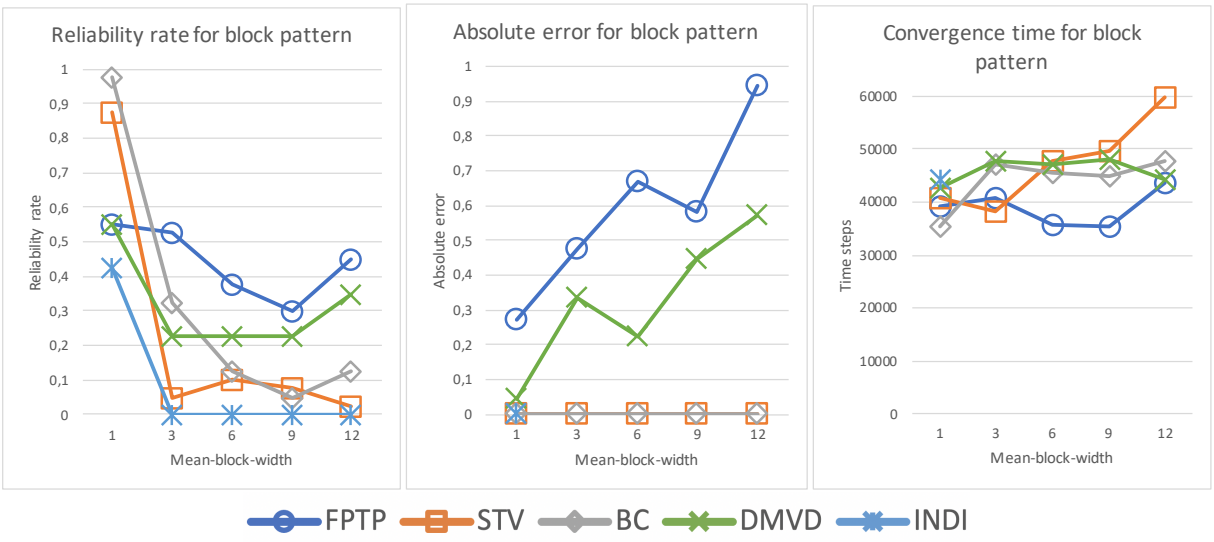

Figure 4.5: Reliability rate, absolute error and convergence time for all 5 analyzed algorithms with 20 runs for each different mean-block-widths for the block pattern including BTRs of 0.35 and 0.45.

results were recorded for the reliability, where we also see the convergence time. FPTP again performed the best with a CT of 39,404.030. The second place is taken by DMVD with 47,571, closely followed by BC and STV as the slowest. These averages only consist of results generated for MBW with 3 to 12 because 1 is not a true block pattern.

Next, we compared the performance of the algorithms for the different MBWs. An underlying visualization of the data the comparison is based on can be seen at the diagrams in 4.5.

FPTP stands out with an almost linear negative trend for accuracy. The reliability value of 0.56 should serve as a guide for an MBW of 1. For the highest MBW of 12, we recorded a rate of 0.41. We record the following changes from 1 to 12: 0.4, 0.4, 0.5, and 0.2. FPTP also shows an upward trend in absolute error and a downward trend in accuracy. These values confirm our assumption. The increase for the MBW of 9 can be attributed to the fact that the generated block has the fewest artifacts at this value. DMVD, which is very similar to FPTP, shows a similar but more moderate behavior with better values and also a fluctuation for the MBW of 6. This result confirms our assumption that the swings are due to the random components of the algorithm. On the other hand, we could not detect any trend in the CT of the FPTP strategy. At the beginning of the clustering (i.e., the

experiment where we go from an MBW of 1 to 3), we registered an expected increase in the time the algorithm needs to find consensus. This value then drops from 3 to 6 to increase again from 9 to 12. A trade-off for FPTP is not apparent.

The data generated by the STV algorithm in the benchmarks with the different MBW show trends but do not generally confirm our assumptions. From an MBW of 1 to 3, the reliability drops by 0.825, which marks the point of the first clustering of black tiles. The measured values continue to fall, reaching a low of 0.05 at 9. Then they unexpectedly rise again by 0.075, contradicting our assumption of a constant negative trend. As before, this can be explained by the use of ranking and the fact that STV does not only rely on the currently executed decision of the agent but on a more complex voting procedure, as explained in Chapter 3.
For the CT, we recorded values, which agree mostly with our expectations except for the change of -5,9% from the MBW of 1 to 3. However, the values increase again, which shows that with the increasing size of blocks, the algorithm takes longer to find a consensus. The difference from 3 to 12 is 21,450, which is the largest increase in time required in our tests. STV has the worst overall performance considering convergence time when confronted with the block pattern. The agents take, on average, 59,801 steps to find consensus in such an arena. This value is just below 60,000 steps, which represents the mark where the runs are stopped.

The reliability of BC has a steady negative trend for the MBW from 3 to 9, falling by 0.275 overall, which would confirm our hypothesis. However, from 9 to 12, the curve shows an improvement of 7.5%, which contradicts our conjecture. This result can be explained by the fact that at an MBW of 9, according to our setup, the environment forms a black block with only a few irregularities. The irregularities at the edge can be interpreted as noise or artifacts, and they change the qualities of the hypotheses of agents perceiving them. These effects cause them to become more uncertain about their own opinions and introduces diversity into the ranking of agents. It is worse for the scoring system when two participating agents both have exactly opposite rankings. This is made less likely by the artifacts than if only perfect blocks existed. The scoring system takes advantage of this behavior, which leads to a greater chance of consensus than with totally

mismatched rankings. A notable correlation can be seen here with DMVD. Between 6 and 12, BC and DMVD have almost the same curve for reliability. However, since there is hardly any correlation between these two strategies except for the effect of the qualities of the advertised decision on the length of the dissemination time we assume that this is a coincidence. This assumption is additionally reinforced by the fact that we have also not seen any correlations in the previous tests.
Similar to STV, BC's accuracy is extremely robust to the block pattern. We also attribute this to the ranking system and, in the case of BC, to the fact that it builds consensus among all participating agents with each vote instead of focusing on the majority.
The CT shows significantly worse values at the beginning for the block pattern compared to the random pattern. We recorded an increase of 33% for 3 in contrast to an MBW of 1. However, this value then falls again and forms a slight parabolic curve with only small deviations with a maximum of 7,8% from 9 to 12. BC has the slowest runs at an MBW of 12. In this case, the algorithm needs, on average, 47,721 steps to find a consensus. If we do not attribute these changes to random fluctuations, we must assume that our previous estimation, in this case, was not correct. When changing from an MBW of 1 to 3, the black squares start to cluster. At 3, several smaller blocks have formed that overlap in some places. This overlapping can make it harder for the agents to find a consensus when the agents are distributed unfavorable across the arena. For MBWs of 6 and 9, the resulting pattern most closely resembles a large block. The larger the blocks and the clearer their shape, the faster the quality determination. This leads to slightly faster convergence of the swarm for 6 and 9.

The gathered data let us conclude that the changed arrangement of the different colors in the environment also has a significant impact on our implemented algorithms. The $\overline{AE}$ for STV and BC is still 0 and shows that even in difficult environments, the swarm makes the correct decision without error, even if only a few converge at all. FPTP turns out to be relatively robust in terms of reliability and CT, but unfortunately also inaccurate. We recognize that the algorithms react differently to the size and completeness of the blocks, and in this respect, we assume that 9 represents the "best" block. FPTP is fast for clear blocks but less accurate and more unreliable. STV becomes much slower as block size increases but has higher reliability on good blocks with fewer artifacts while BC

gets slower and less reliable when the blocks have less noise.

5 Conclusion

This chapter deals with the summary of the analysis of the data we generated. Here we give another short overview, highlight the most important results and what conclusions we can draw from them. The reader is given a well-founded recommendation whether the algorithms we implemented are suitable to use as decision-making in a swarm. The final part is the outlook on possible future research and answers the question how to continue this work.

5.1 Results and Summary

In this thesis, we implemented three algorithms that were inspired by three different majority voting systems. These voting systems are typically used for democratic elections. We designed three algorithms that are based on them and implemented them as the DM of a swarm dealing with a best-of-n problem. The swarm was tasked with finding the correct ratio of black tiles in an environment consisting of black and white tiles out of a list with possible decisions. Therefore, we implemented the first-past-the-post, the single transferable vote, and the Borda count algorithm. We have argued that a DM strategy relying on the majority will be a suitable alternative approach to be used effectively in a swarm.

To determine the performance of the algorithm and the quality of our statement, we tested the approaches several times with different settings in different environments. In doing so, we increased the expressiveness and quality of our benchmarks. To do that, we changed the ratio of black tiles, the time interval of the given states, and the pattern

formed by the colored tiles in the arena. The performance of the algorithms was measured according to three different metrics. First, we analyzed how many samples of the experiments have converged in relation to how many did not. This metric represents the reliability rate and provides us information about how likely it is that the swarm will converge at all. The accuracy is based on the absence of an absolute error. For this metric, we compared the actual collective decision with the correct decision of the sample. We then calculated the mean of all converged samples for the given configuration. We also have chosen the time until a swarm converges as a metric, which we called convergence time. We compared the data of our approaches with two additional collective DM strategies to their performance finding the correct decision.

Our FPTP approach for collective DM shows some trends when tested with black-tile-ratios (BTRs) between 0.05 and 0.45. The reliability decreases with higher BTR settings, but the accuracy increases. The time it needs to convergence is rather robust to BTR changes. While it was tested with different time intervals, there were similar changes as before. When the interval time (IT) raises from 1 to 10, the reliability rate shrinks, but it becomes more accurate. The CT declines with higher ITs. This result shows us that it is not robust to time interval changes. Finding the correct time interval is difficult because the trade-offs need to be balanced. When we compare the performance of FPTP on a random pattern to its performance on a block pattern, we can clearly see notable differences. The reliability rate and the CT on a random pattern are better, but the accuracy is lower. With an increasing mean-block-width (MBW) of the blocks formed by the random pattern, the FPTP strategy becomes more inaccurate. When the blocks are clearer and have fewer artifacts, it becomes less reliable but converges faster. Out of this data, we can conclude that FPTP is environment-sensitive. In general, we found that FPTP has strong trends and trade-offs, mostly between reliability and accuracy. While it is the fastest of our algorithms and has mediocre reliability, its poor accuracy disqualifies it, in our opinion, as a useful collective decision-making approach. For the algorithm based on STV, we recorded an absolute error of 0 for all of our experiments for random and for the block pattern. These results mean it has perfect accuracy, and it always finds the correct consensus. STV shows strong trends when we focus on the curve across the BTR settings. The reliability rate has a negative trend but still performs well overall. The convergence

time (CT) also has a negative curve with decreasing gradients. When looking upon the performance of STV for different time interval settings in a random pattern, we can see that it performs quite well and that the reliability rate does not rely on the duration of the state durations. The same goes for the CT. We can see some fluctuations, but those are caused by the random nature of the algorithm. While the results of STV for a random pattern were promising, the performance declines strongly when tested on a block pattern. Its reliability rate is the worst of all the tested algorithms with independent voting (INDI) as an exception, which did not converge at all, and the CT was the highest. Thus, it was the least reliable and the slowest of all approaches tested on a block pattern. With an increasing mean width of the blocks, it is slower, but for blocks with less artifacts, it gets a bit more reliable. STV is a good choice to take when we can be sure that the environment seems to be a random pattern. It is quite reliable and has perfect accuracy. Nevertheless, we would not recommend using it when the possibility of block clustering exists. Our BC approach has the same perfect accuracy as STV, with an average absolute error of 0 for all tested settings. Its reliability also shows the highest robustness to changes of the BTR in the environment but still has a negative trend. These environment changes have less impact on the reliability rate of BC than it has on STV, but it is not as robust in terms of CT as FPTP. Both the reliability and the CT do not show much sensitivity to changes in the duration of the states. We can notice some fluctuations that are not correlated to the time interval changes. Similar to the STV approach, our BC strategy has significant losses of performance when the colored tiles cluster. It is less reliable, and the swarm needs longer to find the convergence. For the different MBW, there is no remarkable trend. It performs faster but less reliably for blocks with an MBW of 6 and 9. These settings lead to better blocks with fewer artifacts. In general, we can conclude that BC is the best of our three implemented and tested algorithms. It has perfect accuracy, is robust to most changes, and has no remarkable trade-offs. The values and the performance were the best in most of our tests. However, we also found a strong dependence on the environment. While it contains its accuracy for the block pattern, it becomes slow and unreliable. Therefore, it is again only recommended for a predictable environment, but it is, in our opinion, a better option than STV and FPTP. If we now try to find the best of our three algorithms, there is no algorithm that beats the other two in all aspects. We have to decide which qualities are most important to us. While

FPTP is the fastest in a random environment, BC is the most reliable and, together with STV, significantly more accurate. Furthermore, compared to BC and STV, FPTP shows a reduced sensitivity to the block pattern and even shows the highest reliability there. BC seems to be the best choice. Its quality, which decreases for the block pattern, is balanced by the high overall performance and the consistently perfect accuracy, which is what makes it the best choice, in our opinion. The reason for these results is that they consider all possible decisions and each of the agents' rankings to find a compromise each voting round. DMVD is a better alternative to FPTP. It is not as fast but more accurate, which balances it. After the final analysis of the data, we can say that our metrics and settings have been chosen well. They were expressive enough, easy to compare, and interpretable.

5.2 Future Research

As mentioned in Chapter 2, the BC voting system qualifies as a consensus-based voting system. In order to follow up this approach, it would be useful to test more consensus-based voting systems rather than majority voting systems in some future experiments. However, there are many more different majority voting algorithms, which are used for election and democratic voting. Future research is needed to investigate if there are more suitable alternatives to the collective DM in a swarm for a best-of-n problem or if this field will not bring up other useful options.

There are several ways how we could improve the results in further experiments. We could increase the number of agents used in the swarm and increase the time until a test is aborted. We used a threshold of 0.6 for the STV voting. Varying this threshold would be a suitable way for trying to improve the performance of the algorithm. While we have tested our approaches for a random pattern and block pattern, there are several other patterns proposed by Bartashevich et al. [2]. Using them to benchmark the FPTP, STV, and the BC algorithm would improve the expressiveness of the benchmark, and it might bring up some new knowledge about the behavior of the swarm. We used a time interval of 8 seconds for the test with the block pattern. We decided it was the best option, but

in future works, the effect of the time interval on different patterns should be analyzed. We must mention again that the STV is used to fill up a number of seats in a government instead of finding one winner. Future research could examine if the STV would be more effective in a centralized approach, where the agents vote for seats of the decisions with the higher rankings instead of only bringing up one winner. In addition, an implementation of STV and BC with unbiased state duration instead of using influenced state times would be an interesting addition for future research.

Bibliography

[1] Hazem Ahmed and Janice Glasgow. “Swarm intelligence: concepts, models and applications”. In: *School Of Computing, Queens University Technical Report* (2012).

[2] Palina Bartashevich and Sanaz Mostaghim. “Benchmarking collective perception: new task difficulty metrics for collective decision-making”. In: *EPIA Conference on Artificial Intelligence*. Springer. 2019, pp. 699–711.

[3] Eric Bonabeau, Directeur de Recherches Du Fnrs Marco, Marco Dorigo, Guy Théraulaz, Guy Theraulaz, et al. *Swarm intelligence: from natural to artificial systems*. 1. Oxford university press, 1999.

[4] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. “Swarm robotics: a review from the swarm engineering perspective”. In: *Swarm Intelligence* 7.1 (2013), pp. 1–41.

[5] Arne Brutschy, Alexander Scheidler, Eliseo Ferrante, Marco Dorigo, and Mauro Birattari. ““Can ants inspire robots?” Self-organized decision making in robotic swarms”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 4272–4273.

[6] Zhihua Cui and Xiaozhi Gao. *Theory and applications of swarm intelligence*. 2012.

[7] M Dorigo et al. “Blockchain technology for robot swarms: A shared knowledge and reputation management system for collective estimation”. In: *Swarm Intelligence: 11th International Conference, ANTS 2018, Rome, Italy, October 29–31, 2018, Proceedings*. Vol. 11172. Springer. 2018, p. 425.

[8] John Dunn. “Democracy: The unfinished journey”. In: (1995).

[9] Julia T Ebert, Melvin Gauci, and Radhika Nagpal. “Multi-feature collective decision making in robot swarms”. In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 2018, pp. 1711–1719.

[10] MDLM Gambardella, Mauro Birattari Alcherio Martinoli, and Riccardo Poli Thomas Stützle. “Ant colony optimization and swarm intelligence”. In: *5th international workshop, Springer*. Springer. 2006.

[11] Aleksandar Jevtić and Diego Andina de la Fuente. “Swarm intelligence and its applications in swarm robotics”. In: (2007).

[12] David Lippman. “Voting theory”. In: *Creative Commons BYSA* (2013).

[13] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. “The e-puck, a robot designed for education in engineering”. In: *Proceedings of the 9th conference on autonomous robot systems and competitions*. Vol. 1. CONF. IPCB: Instituto Politécnico de Castelo Branco. 2009, pp. 59–65.

[14] Giuseppe Morlino, Vito Trianni, Elio Tuci, et al. “Collective Perception in a Swarm of Autonomous Robots.” In: *IJCCI (ICEC)*. 2010, pp. 51–59.

[15] Chris AC Parker and Hong Zhang. “Biologically inspired collective comparisons by robotic swarms”. In: *The International Journal of Robotics Research* 30.5 (2011), pp. 524–535.

[16] Kevin M Passino and Thomas D Seeley. “Modeling and analysis of nest-site selection by honeybee swarms: the speed and accuracy trade-off”. In: *Behavioral Ecology and Sociobiology* 59.3 (2006), pp. 427–442.

[17] Judhi Prasetyo, Giulia De Masi, and Eliseo Ferrante. “Collective decision making in dynamic environments”. In: *Swarm Intelligence* 13.3 (2019), pp. 217–243.

[18] Louis Rosenberg. “Artificial Swarm Intelligence, a Human-in-the-loop approach to AI”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016.

[19] Louis Rosenberg, Niccolo Pescetelli, and Gregg Willcox. “Artificial Swarm Intelligence amplifies accuracy when predicting financial markets”. In: *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. IEEE. 2017, pp. 58–62.

[20] Nathan Russell. “Complexity of control of Borda count elections”. In: (2007).

[21] Alexander Scheidler, Arne Brutschy, Eliseo Ferrante, and Marco Dorigo. “The k-Unanimity Rule for Self-Organized Decision-Making in Swarms of Robots”. In: *IEEE transactions on cybernetics* 46.5 (2015), pp. 1175–1188.

[22] Thomas D Seeley and Susannah C Buhrman. “Group decision making in swarms of honey bees”. In: *Behavioral Ecology and Sociobiology* 45.1 (1999), pp. 19–31.

[23] Thomas D Seeley, P Kirk Visscher, and Kevin M Passino. “Group Decision Making in Honey Bee Swarms: When 10,000 bees go house hunting, how do they cooperatively choose their new nesting site?” In: *American scientist* 94.3 (2006), pp. 220–229.

[24] Bongrae Seok. *Collective Decision Problem*. Ed. by Deen K. Chatterjee. Dordrecht: Springer Netherlands, 2011, pp. 155–156. ISBN: 978-1-4020-9160-5. DOI: 10.1007/978-1-4020-9160-5_141. URL: https://doi.org/10.1007/978-1-4020-9160-5_141.

[25] Qihao Shan and Sanaz Mostaghim. “Collective Decision Making in Swarm Robotics with Distributed Bayesian Hypothesis Testing”. In: *International Conference on Swarm Intelligence*. Springer. 2020, pp. 55–67.

[26] Volker Strobel, Eduardo Castelló Ferrer, and Marco Dorigo. “Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario”. In: (2018).

[27] Nicolaus Tideman. “The single transferable vote”. In: *Journal of Economic Perspectives* 9.1 (1995), pp. 27–38.

[28] Gabriele Valentini, Davide Brambilla, Heiko Hamann, and Marco Dorigo. “Collective perception of environmental features in a robot swarm”. In: *International Conference on Swarm Intelligence*. Springer. 2016, pp. 65–76.

[29] Gabriele Valentini, Heiko Hamann, and Marco Dorigo. “Efficient decision-making in a self-organizing robot swarm: On the speed versus accuracy trade-off”. In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. 2015, pp. 1305–1314.

[30] Gabriele Valentini, Heiko Hamann, Marco Dorigo, et al. “Self-organized collective decision making: the weighted voter model.” In: *AAMAS*. 2014, pp. 45–52.

[31] Jing Wang and Gerardo Beni. “Cellular robotic system with stationary robots and its application to manufacturing lattices”. In: *Proceedings. IEEE International Symposium on Intelligent Control 1989*. IEEE. 1989, pp. 132–137.

Statement of Authorship

I herewith assure that I wrote the present Master thesis independently, that the thesis has not been partially or fully submitted as graded academic work and that I have used no other means than the ones indicated. I have indicated all parts of the work in which sources are used according to their wording or to their meaning.

I am aware of the fact that violations of copyright can lead to injunctive relief and claims for damages of the author as well as a penalty by the law enforcement agency.

Signature

Date