POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Collective resilience of heterogeneous decision makers against stubborn individuals

Tesi di Laurea Magistrale in
Computer Science and Engineering - Ingegneria Informatica

Author: **Nemanja Antonic**

Student ID: 995553
Advisor: Francesco Amigoni
Co-advisors: Andreagiovanni Reina, Raina Zakir
Academic Year: 2022-23

Abstract

A swarm is a group of agents with local knowledge of the environment, that collaborate, coordinating their actions in order to perform a task. In this study, I address the best-of-n problem, where the objective of the swarm is that of reaching an agreement on the best among n options, each having an associated quality. I focus on the collective decision-making process to reach a collective agreement on the best option, which is guided by the exchange of opinions among the agents. In this thesis, I test the resilience of the swarm when malicious agents, that spread misinformation purposefully, are present. The malicious agents that I consider are stubborn individuals, also named zealots, which never change their opinion regardless of social discordance. Each agent is able to hold one opinion at a time and this opinion may change as a result of social interactions. Moreover, I consider different behavioural configurations of the agents, first with swarms that display homogeneous behaviours, where every agent within the swarm follows the same collective decision-making strategy, and combine them to form heterogeneous swarms where the collective decision-making strategies differ across agents within the swarm. The main difference between these behavioural models is the amount of social information that each agent needs to process, where models that need to process more social information are more resilient; however, processing more information implies a higher cognitive cost. Hence, the focal point of this work resides in the combination of behavioural models that require less social information with others that use more social information to understand the balance between resilience and cognitive cost. I study the performance of such hybrid swarms by building and analysing mathematical models which represent these behaviours. These models consist of a system of ordinary differential equations that describe how the population splits into sub-populations composed of agents holding the same opinion and how the size of these sub-populations evolves over time. I analyze and compare these models for different parameter values, such as different numbers of zealots and qualities of the options. I analyze the models in homogeneous swarms first, in order to understand the baseline behaviour of such systems; then, I combine them into heterogeneous swarms. I develop metrics in order to evaluate the different models in terms of how well they perform in terms of cognitive cost. The results show that heterogeneous models allow to have a

trade-off between performance and cost, while this is not possible in the homogeneous models. Furthermore, heterogeneous models allow to fine tune the number of agents that require less social information in order to meet specific requirements in terms of resilience and cognitive cost.

Keywords: collective resilience, collective decision making, swarm robotics, group heterogeneity

Abstract in lingua italiana

Uno sciame è un gruppo di agenti con conoscenze locali dell'ambiente, che collaborano coordinando le loro azioni al fine di svolgere un compito. In questo lavoro, mi concentro sul *best-of-n problem*, dove l'obiettivo dello sciame è il raggiungimento di un accordo sulla migliore tra n opzioni, ognuna delle quali ha una certa qualità. Mi concentro sul processo decisionale collettivo che porta al raggiungimento di un accordo comune sull'opzione migliore, il quale è guidato dallo scambio di opinioni tra gli agenti. In questa tesi, svolgo dei test sulla resilienza dello sciame quando sono presenti agenti malevoli, che disseminano disinformazioni intenzionalmente. Gli agenti malevoli che considero sono individui testardi, chiamati anche zeloti, che non cambiano mai la loro opinione nonostante la dinamica sociale. Ogni agente è in grado di avere un'opinione alla volta e questa può cambiare come risultato di interazioni sociali. Inoltre, considero diverse configurazioni comportamentali degli agenti, prima con sciami che mostrano comportamenti omogenei, dove ogni agente nello sciame segue la stessa strategia decisionale, e li combino per creare sciami eterogenei, dove le strategie decisionali sono diversificate all'interno dello sciame. La differenza principale tra questi modelli comportamentali è la quantità di informazioni sociali che ogni agente deve elaborare, dove i modelli che necessitano più informazioni sociali sono più resilienti; tuttavia, elaborare più informazioni implica avere un costo cognitivo più alto. Quindi, il punto principale di questo lavoro risiede nella combinazione di modelli comportamentali che richiedono meno informazioni sociali con altri modelli che utilizzano più informazioni sociali per trovare l'equilibrio tra resilienza e costo cognitivo. Studio la prestazione di questi sciami ibridi costruendo e analizzando modelli matematici che rappresentano questi comportamenti. Questi modelli sono composti da sistemi di equazioni differenziali ordinarie che descrivono come la popolazione si divide in sottopopolazioni composte da agenti che hanno la stessa opinione e come le dimensioni di queste sottopopolazioni evolvono nel tempo. Analizzo e comparo questi modelli per diversi valori dei parametri, come diversi numeri di zeloti e qualità delle opzioni. Analizzo prima i modelli negli sciami omogenei, al fine di comprenderne le proprietà; successivamente, li combino in sciami eterogenei. Inoltre, sviluppo metriche al fine di valutare la loro qualità e il loro costo cognitivo. I risultati mostrano che i modelli eterogenei consentono

un compromesso tra prestazioni e costi, il che non è possibile nei sistemi omogenei. Per giunta, i modelli eterogenei permettono di regolare il numero di agenti che richiedono meno informazione sociale al fine di soddisfare requisiti in termini di resilienza e costo.

Parole chiave: resilienza collettiva, processo collettivo decisionale, sciame di robot, eterogeneità di gruppo

Contents

1 | Introduction

The following work is done in collaboration with the IRIDIA laboratory at the Université Libre de Bruxelles and under the supervision of Prof. Francesco Amigoni, Prof. Marco Dorigo, Andreagiovanni Reina and Raina Zakir.

1.1. Overview

1.1.1. Swarm

The concept of swarm comes from the behaviour of social animals that cooperate in order to solve problems. Such animals display strong grouping behaviours [9], fostering the communal solution of problems. More generally, a swarm can be either natural or robotic: the former comprise the aforementioned animals, such as social insects like termites, ants and honeybees; the latter comprise autonomous robots that communicate with each other, emulating the behaviour of natural swarms. The behavioural inspiration to engineer robotic swarms is driven by the fascinating capability of natural swarms to solve complex tasks in spite of the inherent difficulties and imperfections of such groups: each individual has noisy, imperfect and local data about the environment and the transmission of this data is not free from noise and uncertainty either. Thus, natural swarms are robust to noise, scalable in terms of the number of individuals and flexible towards changes in the environment [3], all properties that are desirable even in autonomous robotic systems. In particular, swarm robotics abides the following criteria, first defined in [3]:

- it is composed of a large number of robots;
- it is composed of a few homogeneous groups of robots;
- the robots are unable to tackle a specific problem on their own, however their performance increases when they collaborate;
- the used robots have limited and local sensing capabilities.

These properties are desirable and allow to emulate natural swarms. To make an example,

I consider honeybees: the swarm is indeed comprised of a high number of individuals and there are a few homogeneous groups, such as cell cleaners and nurses [8]. Finally, they cannot survive alone, let alone solve tasks, and each honeybee has limited and local sensing abilities. However, among the tasks they solve, an example is the nest-site selection [11], where the swarm needs to agree on the best site where to build a nest among a set of alternatives on which the survival of the swarm lies. Even when faced with such challenge, the honeybees as a collective are almost always able to collectively choose the best option for their next nest [15].

1.1.2. Collective decision making

Collective decision making is the process through which a swarm reaches an agreement [1]. The main feature of these groups is that of having no centralised authority: each individual expresses a preference, then these preferences are compared by other individuals, leading to the majority of individuals adopting the same preference. This is done through the use of a set of simple rules that allows each individual to update its opinion, based on the preferences of the others. Within the field of swarm robotics, collective decision making is used whenever the robots need to accomplish a task, which can be formalized as a problem. In order to understand if the task has been successfully solved or not, there needs to be some kind of agreement among the robots. Moreover, collective decision making focuses on the optimal way to reach consensus, by viewing the interactions among the individuals through an economical lens [1], where the decision process is treated as an exchange of goods, causing gains and losses to the group based on the optimality of the process itself. Therefore, the optimality of the process is guided by the maximization of the gains and the minimization of the losses.

1.1.3. Best-of-n problem

The best-of-n problem, in the context of collective decision-making, is a specific class of problems, where a group of agents need to come to an agreement on the best option, among n options. In order for an option to qualify as the best, each option has an associated quality, which measures numerically the benefit the group grains from choosing it, in particular in terms of how important that option is for the agents. Following the taxonomy defined in [18], this kind of problem is a discrete consensus achievement problem. It is discrete, because the number of options is a natural number. In particular, I use two options, thus $n = 2$. It is a consensus achievement problem because it requires the majority of the agents to agree on the best option, where best is related to the needs of the agents.

1.1.4. Stubborn individuals

The concept of stubborn individual is quite general, as individuals that do not change their opinion, despite the social information that is available to them, exist in different contexts. In the context of collective decision-making, these individuals constitute non-collaborative agents, since they do not update their opinion, despite the information that collaborative agents provide to them. From a practical perspective, stubborn individuals may represent faulty robots, that have, for example, a broken sensor, thus they are not able to receive nor process messages. Moreover, stubborn individuals may be malicious agents whose purpose is that of swaying the collaborative population away from the option that best suits the needs of the group. Such agents are also called zealots and they are byzantine agents, since collaborative agents cannot distinguish them from other collaborative agents. If the objective of the swarm in the best-of-n problem is that of reaching consensus on the best option by exchanging their local information about the quality of the options and subsequently adapt their belief on the best option, stubborn individuals never change their belief and thus always share the same opinion.

1.1.5. Resilience

Resilience, in the context of swarm robotics, refers to the ability of a system to reach its goal despite the presence of some kind of disturbance. The latter has an impact on the swarm and the response of the system needs to be tested against different values of the disturbance. The robots already suffer from noise and local, hence incomplete, information, thus this additional disturbance configures a test on the ability of the swarm to adapt, challenging its flexibility. In this work, the disturbance is represented by the presence of stubborn individuals, that do not change idea despite the social information provided to them. I test the behaviour of the system with different amounts of stubborn individuals, seeking how well it behaves and the required computational cost. In particular, I underline the presence of a trade-off between the expected behaviour and the cognitive cost, a measure of how many messages a model needs to process, associated to different models. This trade-off is a natural consequence of the presence of a disturbance and its magnitude influences the choice of optimal design parameters.

1.2. Problem formulation

In this thesis, I develop and analyze four heterogeneous models that represent the collective decision-making process of a swarm. The problem I tackle is finding the best trade-off

between accuracy and cognitive cost of heterogeneous models with different percentages of agents using a simple opinion filtering mechanism, while the rest of the population uses a more complex mechanism. These mechanisms define how an agent chooses an opinion based on the messages it receives from its neighbours: the simpler mechanisms is the voter model, where an agent chooses a random opinion from the messages of its neighbours, while the more complex one is the majority rule, where an agent chooses the opinion contained in the majority of the messages of its neighbours. Indeed, this research is guided by the idea that the combination of different opinion filtering mechanisms within the swarm allows us to balance both accuracy and cognitive cost, unlike the homogeneous swarm where this is not possible. Moreover, I add stubborn individuals to the swarm, whose objective is that of either swaying the collaborative population away from the optimal solution, or that of making the agreement of the collaborative population unfeasible, by inducing a decision deadlock. Thus, the problem can be formalized as:

Let k be the percentage of agents using the voter model and let $1-k$ be the percentage of agents using the majority rule in a collaborative population. Assume an unknown amount of stubborn agents is added. Let accuracy be the metric that measures how well a model performs in terms of how often the agents agree on the best option. Furthermore, let the cognitive cost be the metric that measures the number of messages needed by a model to reach consensus. Then, can there be a $k \in (0,1)$ such that a heterogeneous model with k agents using the voter model has an optimal trade-off, in terms of performance and cognitive cost?

1.3. State of the art

This work is deeply rooted in both swarm robotics and collective decision-making. These fields have different approaches in the analysis of the resilience of swarms to stubborn individuals and, more in general, to all types of malicious agents. In [13], Reina et al. analyse the resilience of the cross-inhibition opinion update rule, compared to the direct switch rule. Their results show that cross-inhibition is able to break decision deadlocks more easily than direct switch, which is in line with the analysis I conduct on the homogeneous models. The decision-making process applied to a finite set of options is studied in [12], where the authors explore the influence of spatiality in the best-of-n problem. The authors in [2], exemplify the state of the art in opinion dynamics through statistical physics. In particular, they tackle the voter model and the majority rule, which I use in my work. The best-of-n problem is formally defined in [18], where the authors shed light on certain

variants of this class of problems, describing a taxonomy that differentiates various types of best-of-n problems. The authors in [6] create a model-independent framework in order to study the dynamics of decision-making processes and then in [7] it is further analyzed and applied to a multi-robot task-allocation problem, however they do not consider the presence of malicious agents and thus the resilience is not studied. In [11], the authors provide a design pattern for decentralised decision making, through the modeling of both homogeneous and heterogeneous swarms, from a microscopic level, in such a way that it matches with the macroscopic level behaviour. In fact, my work matches their general design pattern, but my analysis focuses more on the specific behavioural differences that define how an agent chooses an opinion starting from the messages it receives.

Resilience is studied under a time-varying communication topology in [22]. In [16], the authors define a blockchain-based technology that allows for resilience against byzantine robots. Furthermore, the authors in [21] test the resilience of a swarm in a best-of-n problem with $n > 2$ by using opinion dynamics. The authors in [20] explore the effects of zealots on binary opinion dynamics and show its effects. However, none of these works related on the resilience of swarms against stubborn agents implements the idea of a heterogeneous swarm.

The authors in [19] define a resilient heterogeneous model for underwater robotic swarms. Furthermore, the authors in [4] explore the concept of heterogeneity in swarm robotics, by claiming its importance as a tool to confront complexity. Moreover, in [5], the authors define a heterogeneous swarm that tackles the solution of a task-allocation problem, with some robots that act as visual explorers and other robots that perform actual foraging. Nonetheless, these works do not analyze the resilience of the modelled systems.

In [17], the authors explore the trade-off between speed and accuracy in a binary discrimination problem, when using the majority rule. They find that the main factor affecting this trade-off is the number of agents that take part in the voting, however, they do not take into account the possibility of having a heterogeneous swarm. In fact, I show that the trade-off can be optimally set solely by choosing the percentage of agents using the voter model.

1.4. Original contribution

The following work proposes the following original contributions:

- the extension of homogeneous systems of collective decision-making for the best-of-2 problem to heterogeneous swarms, where the population is split into two sub-populations following different collective decision-making behaviours and interacting with each other;
- the exploration of a large parameter space of the ODE models to determine the outcome of a decision-making process, which allows to compare the homogeneous and the heterogeneous models;
- a method to evaluate the trade-off between the accuracy, which is related to the resilience of the model, and the cognitive cost associated to a specific model, which allows to compare and rank hybrid models in terms of these metrics.

Moreover, this thesis leverages the following results:

- the definition of the $P_\alpha(x)$ function used in Section 2 by the authors in [14], which allows to define the probability of receiving a message with a particular opinion;
- the definition of the discrete integration of a Bernoulli distribution to find the probability of receiving an opinion, which I have re-worked, first defined in [17];
- the regret metric, described in [10], which I have adapted to this study to account for the value lost by the group in terms of the differences in qualities of the options.

1.5. Structure

The following thesis is organized as follows: in Chapter 2, I describe the assumptions and the mathematical requirements of the filtering mechanisms and of the update rules. Moreover, I develop the systems of ordinary differential equations that allow to describe both homogeneous and heterogeneous swarms. Finally, I formalize the metrics to evaluate and analyse the aforementioned models.
In Chapter 3, I apply those metrics to four heterogeneous systems, which are a more general case of the homogeneous systems, hence I am able to evaluate their trade-off in terms of accuracy and cognitive cost, together with the other metrics.
Finally, the conclusion and the future works are explained in Chapter 4, where I discuss the use of simulations and of robot experiments.

2 | Models

Swarm robotics is a field of robotics in which a group of robots, a swarm, collaborates in order to solve a problem. The robots communicate with each other, sharing their current opinion on the best option, and they explore the environment, collecting information about the quality of the options they encounter. All the information they collect and exchange is inherently local, as no robot has complete knowledge of the environment. The communication of the robots is then related to the exchange of the local information they collect, in order to make decisions through a collective decision-making algorithm. Often in collective-decision making, the swarm needs to adopt the option with highest quality among n possible options, each with its own quality. The goal of the robots is then to reach a consensus on the best option, which means that the majority of the swarm needs to hold that option. I analyze and model the opinion dynamics surrounding collective decision making process in swarm robotics, focusing on the social behaviour of the robots, which I refer to as agents.

In order to solve a best-of-n problem in collective decision-making, all agents will first apply a filtering mechanism to choose one message among those sent by its neighbours. The agent uses an opinion selection mechanism to filter the received information and an option update rule to update its own opinion. Figure 2.1 shows this process from the point of view of the single agent. This chapter is dedicated to the development of the mathematical models of the update rules and of the filtering mechanisms in best-of-n decision problems, which I adapt from [11], [14] and [17] to account for the presence of zealots. To set a common notation among all the models, let there be two options, namely option A and option B, each with its own quality, q_A and q_B, respectively. Therefore, in the following, I handle a best-of-n with $n = 2$ problem, where the option with the highest quality is option A. Its quality is fixed to 1, which implies that

$$0 \leq q_B \leq 1 \tag{2.1}$$

The agents communicate with a frequency that is proportional to the quality of their

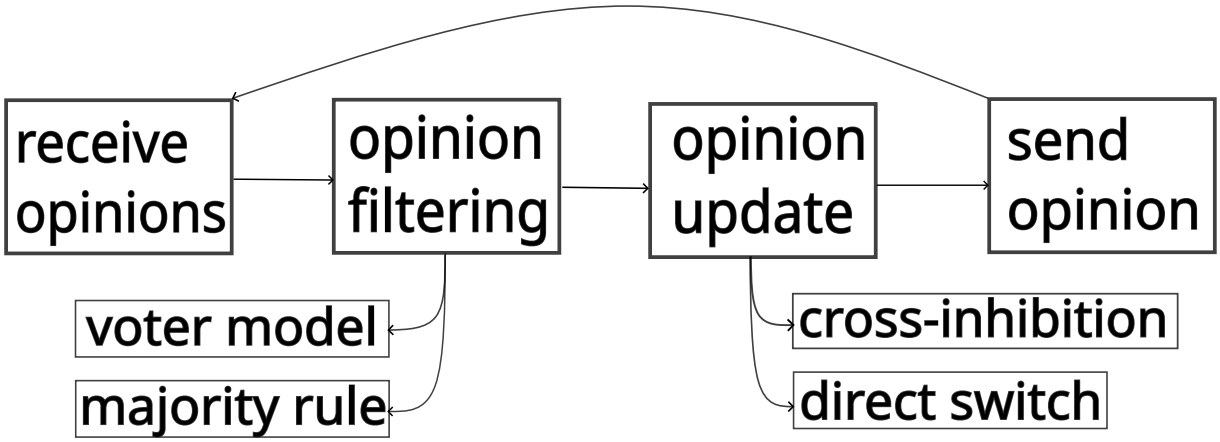


Figure 2.1: The process each agents undergoes at every iteration: first, it receives the opinions from it neighbours; then, starting from those, it picks an opinion that is applied to its opinion update rule. Finally, the agent disseminates the opinion.

option [18], a behaviour that has been observed also in honeybees. In an optimal setting, all the agents are collaborative, which means that their collective decision-making efforts are in line with the goal of agreeing on the best option. However, the possibility of having also non-collaborative agents, or malicious agents, exists. Thus, let the size of the collaborative population be fixed to some $N \in \mathbb{N}$ and let the number of collaborative agents who hold opinion A (respectively, opinion B) at time t be $n_A(t)$ ($n_B(t)$). These quantities depend on time since they change as the system evolves. What I consider constant is the number of collaborative agents, which implies that the robots do not break down while the system evolves, and no new robots join the swarm during the decision-making process. Then, I introduce a separation between collaborative and non-collaborative agents, so that the amount of malicious agents is easily adjustable. In particular, the malicious agents represent zealots, agents that do not change their opinion either because they do not work properly (e.g. their sensors do not function properly, making it impossible for them to receive new opinions) or because they are hostiles that purposely interfere with the collaborative population's task. In the following, I analyze the case of malicious agents, as a pessimistic assumption. Therefore, zealots communicate with maximum frequency (equivalent to q_A) independently of the quality of the opinion they hold. The definition of the number of zealots then is related to N. In particular, I express the number of zealots Z as a percentage normalised on N, with z_A and z_B indicating the relative amount of zealots holding opinion A and opinion B:

$$Z = N(z_A + z_B) \tag{2.2}$$

The quantities z_A and z_B are considered constant. Let the relative number of collaborative agents holding opinion A (B) be defined as $A(t)$ $(B(t))$:

$$A(t) := \frac{n_A(t)}{N} \tag{2.3}$$

$$B(t) := \frac{n_B(t)}{N} \tag{2.4}$$

The dependence on time is inherited from $n_A(t)$ $(n_B(t))$. The presence of zealots can lead to two main types of attacks:

- **Wrong Addressing**: the attack goal is to drive the agents to choose the worst option, and so the zealots in the swarm support the inferior quality option ($z_A = 0$).
- **Denial Of Service**: the attack goal is to cause a decision deadlock, meaning that the quorum is not reached on either option and the population remains roughly split in half with respect to the opinions they hold.

2.1. Filtering mechanisms

Filtering mechanisms allow an agent to filter the messages it receives from its neighbours before the update rule is applied. This section is dedicated to two filtering mechanisms, both applicable only to the collaborative agents: the voter model and the majority rule. The former consists of an agent picking a message containing an opinion at random from its neighbourhood, while the latter corresponds to an agent picking the opinion which is transmitted by the majority of its neighbours. To capture these two models, let the weighted fractions of agents bearing opinion A and opinion B, denoted by $n_A^{\#}(t)$ and $n_B^{\#}(t)$ respectively, be defined as:

$$n_A^{\#}(t) := \frac{q_A A(t)}{q_A A(t) + q_B B(t)} \tag{2.5}$$

$$n_B^{\#}(t) := \frac{q_B B(t)}{q_A A(t) + q_B B(t)} \tag{2.6}$$

which, by setting $q_A = q_{max} = 1$ and by defining

$$q := \frac{q_B}{q_A}\,, \tag{2.7}$$

become:

$$n_A^{\#}(t) = \frac{A(t)}{A(t) + qB(t)} \tag{2.8}$$

$$n_B^{\#}(t) = \frac{qB(t)}{A(t) + qB(t)}\,. \tag{2.9}$$

By considering also the malicious agents and by letting them communicate with maximum quality, the probability of receiving a message with a vote for A or B is, respectively,

$$n_A^{\#}(t) = \frac{A(t) + z_A}{A(t) + z_A + qB(t) + z_B} \tag{2.10}$$

$$n_B^{\#}(t) = \frac{qB(t) + z_B}{A(t) + z_A + qB(t) + z_B}\,. \tag{2.11}$$

In the worst-case scenario, zealots will trick the population away from the best option using maximum quality, thus z_B is not multiplied by q. Then, let $P_\alpha(x)$ be defined as follows:

$$P_\alpha(n_i^{\#}(t)) := \begin{cases} -\left(\frac{1}{2} - n_i^{\#}(t)\right)^\alpha 2^{\alpha-1} + \frac{1}{2}, & \text{if } 0 \le n_i^{\#}(t) \le \frac{1}{2} \\ \left(n_i^{\#}(t) - \frac{1}{2}\right)^\alpha 2^{\alpha-1} + \frac{1}{2}, & \text{if } \frac{1}{2} \le n_i^{\#}(t) \le 1 \end{cases} \tag{2.12}$$

This function corresponds to the probability of an agent switching their opinion to another option i, where $n_i^{\#}(t)$ is the weighted fraction of agents holding opinion i (where i = A, B) . Furthermore, it encapsulates different voting mechanisms through the manipulation of the α parameter. In particular, for $\alpha = 1$, I obtain the voter model, and for $\alpha = 0$, I obtain the majority rule. The necessary assumption is that the system is well mixed, in order for the agents to be distributed uniformly in the space with respect to their opinions. The well mixed assumption states that the agents are distributed in the environment uniformly with respect to their opinions. In this case, the modelling of local phenomena captures the same trends as the modelling of global phenomena.

Voter model

The agents communicate one with each other and the exchange of messages is regulated by the voting mechanism at hand. In the voter model, an agent picks a random opinion from its neighbourhood and applies it to its update rule: the probability of receiving a random opinion i from any agent is then just the weighted average of the number of agents holding opinion i multiplied by the corresponding quality. In fact, by setting $\alpha = 1$, $P_\alpha(x)$ simplifies to:

$$P_1(n_i^{\#}(t)) = n_i^{\#}(t) \tag{2.13}$$

Basically, this imposes a proportional relationship between the current number of agents holding a certain opinion i and the probability of receiving a message from an agent bearing opinion i.

Majority rule

In the majority rule, an agent picks the opinion held by the majority of its neighbours and then processes it through its update rule. By setting $\alpha = 0$, $P_\alpha(n_i^\#(t))$ simplifies to:

$$P_0(n_i^\#(t)) = \begin{cases} 0, & \text{if } 0 \leq n_i^\#(t) \leq \frac{1}{2} \\ 1, & \text{if } \frac{1}{2} \leq n_i^\#(t) \leq 1 \end{cases}$$

Here, the probability of receiving opinion i is dictated by the number of agents holding opinion i: if that number is greater than half, then it is safe to say that the agent will pick opinion i; otherwise, the opinion will not be picked. Thus, this probability expresses the likelihood of an agent choosing an opinion when the majority of its neighbours hold that opinion. In order to work with continuous functions, $P_0(x)$ is approximated by a sigmoid function:

$$P_0(x) \approx \frac{1}{1 + e^{\kappa(0.5-x)}}\,, \tag{2.14}$$

with $\kappa \gg 1$, in my case, $\kappa = 100$. Another way to model the majority rule is to consider a fixed group size G and consider all possible combinations of neighbours' opinions to determine the probability of having a majority for an opinion that is different than the current one [17]. The group size is the number of neighbours an agent receives messages from, which, once fixed, makes it possible to write the probability of choosing opinion A (respectively, opinion B) as:

$$p_A(t) = \sum_{i=\lceil \frac{G}{2} \rceil + 1 - G(\mathrm{mod}\ 2)}^{G} \binom{G}{i} [n_A^\#(t)]^i [n_B^\#(t)]^{G-i} \tag{2.15}$$

$$p_B(t) = \sum_{i=\lceil \frac{G}{2} \rceil + 1 - G(\mathrm{mod}\ 2)}^{G} \binom{G}{i} [n_B^\#(t)]^i [n_A^\#(t)]^{G-i}\,, \tag{2.16}$$

where the symbol $\lceil ... \rceil$ indicates the ceiling operator, which approximates a real number to its closest integer such that it is greater or equal than the integer part of the real number, and $\binom{G}{i}$ indicates the Newton binomial, which represents the possible subsets, each composed of i elements, of the set with G elements. The idea is that the probability of choosing a message with an A (respectively, B) opinion is the discrete integration of

a Binomial distribution where the success probability is the probability of receiving a message containing opinion A (B), which is equal to $n_A^{\#}(t)$ ($n_B^{\#}(t)$). The number of trials is G and i is the number of successful tries. A different opinion is taken if and only if the number of neighbours sharing that opinion is greater than half of the total number of neighbours. The modulus two of the group size serves the purpose of dealing with even and odd group sizes:

- if G is odd, the agent needs at least $\lceil \frac{G}{2} \rceil$ neighbours with the same opinion to pick that opinion;
- if G is even, the agent needs at least $\lceil \frac{G}{2} \rceil + 1$ neighbours with the same opinion to pick that opinion.

2.2. Update rules

In the following, I discuss the development of the mathematical models for the two main update rules, direct switch and cross-inhibition. These apply only to the collaborative part of the population.

2.2.1. Direct switch

In the direct switch update rule, an agent switches its opinion whenever it receives an opinion that is different from its current belief. Therefore, an agent may be in either of the two states, namely A and B. I define the update rule as follows, where the received opinion is defined as the opinion that is picked by the filtering mechanism, as in Section 2.1:

```
function direct_switch(received_opinion):
    agent.opinion ← received_opinion
```

In this behaviour, the agent is very susceptible and switches its opinion directly. A finite state automaton showing the way an agent switches its opinion based on the filtered opinion is depicted in Figure 2.2: if the current opinion of an agent is A and the agent filtered opinion from among its neighbours is B, then it switches to B; if the current opinion is B and the filtered opinion is A, then it switches to A. Finally, if the current opinion and the filtered opinion coincide, there is no effect on the current opinion. Since there are only two possible states, the total number of agents in each state is equivalent

to the swarm size:

$$n_A(t) + n_B(t) = N \tag{2.17}$$

Therefore, from Equations 2.3 and 2.4

$$A(t) + B(t) = 1 \tag{2.18}$$

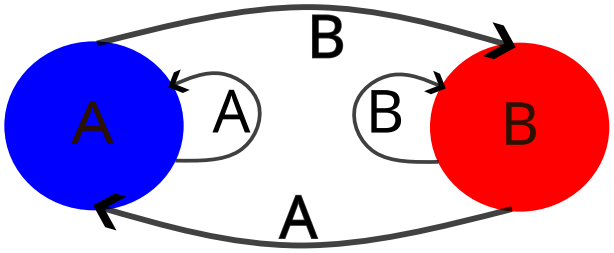


Figure 2.2: Direct switch finite state automaton: an agent holding opinion A switches to B upon filtering a B message and vice versa from B to A upon filtering an A message. The opinion does not switch if the filtered message contains the current agent's belief.

2.2.2. Cross-inhibition

The cross-inhibition update rule introduces a latent state, which corresponds to an agent's opinion being inhibited by a contrasting opinion. In pseudocode, this is equivalent to the following:

```
function cross_inhibition(received_opinion):
if agent.opinion==None then
    agent.opinion ← received_opinion
else if agent.opinion!=received_opinion then
    agent.opinion ← None
end if
```

Thus, an agent can hold opinion A, B or have no opinion. To model this, I introduce a variable $U(t)$, which keeps track of the proportion of agents that are in the uncommitted state (with no opinion) at time t. Since only these three states are available, I state the following:

$$A(t) + B(t) + U(t) = 1 \tag{2.19}$$

where $A(t)$ and $B(t)$ are obtained through the same process as Equations 2.3, 2.4. A finite state machine depicting the cross-inhibition update rule is depicted in Figure 2.3.

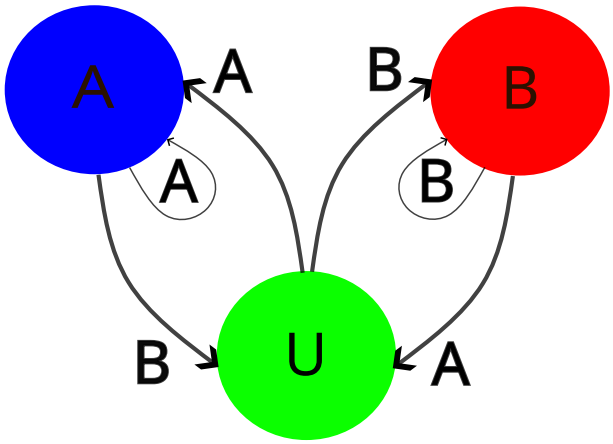


Figure 2.3: Cross-inhibition finite state machine: an agent holding opinion A will switch to U whenever it receives a message with opinion B and vice versa from U to A whenever it receives a message with opinion A; similarly, if it holds opinion B, it switches to U with upon receiving a message with opinion A and vice versa from U to B when it receives a message with opinion B.

2.3. Ordinary Differential Equations

This section is dedicated to the development of the models obtained by combining different update rules and filtering mechanisms. In particular, I model and analyze: direct switch with voter model, direct switch with majority rule, cross-inhibition with voter model and cross-inhibition with majority rule. I construct the models using the majority rule both with P_α (Equation 2.14) and by using the discrete Bernoulli approach (Equations 2.15, 2.16). To build the aforementioned models, I use ordinary differential equations (**ODEs**), together with constraints on the population size. Beyond the homogeneous systems, in which the swarm uses only one model, I also develop heterogeneous systems by properly combining two homogeneous models to form hybrid swarms. These systems take into account the same update rule with two filtering mechanisms, meaning that one part of the population uses the voter model and the other uses the majority rule.

2.3.1. Direct switch with voter model

The Direct Switch with Voter Model (**DSVM** from now on) leverages Equations 2.18 and 2.13. From the former:

$$B(t) = 1 - A(t) \tag{2.20}$$

To follow the evolution of the system, I take into consideration the first derivative of $A(t)$ with respect to time:

$$\frac{dA}{dt}(t) = P_1\left(n_A^{\#}(t)\right)B(t) - P_1\left(n_B^{\#}(t)\right)A(t) = n_A^{\#}(t)B(t) - n_B^{\#}(t)A(t) \tag{2.21}$$

The first term denotes the increase in the A population given by the product of the weighted average of the relative number of agents with opinion A, which is the probability of receiving a message with an A opinion, and the relative number of agents bearing opinion B. This represents the likelihood of changing the opinion of agents with opinion B based on the relative amount of agents with opinion A. In a similar fashion, the second term, with a negative sign, represents the decrease in the A population caused by the propagation of messages with opinion B to agents holding opinion A. Given the dynamic nature of this system, I am interested in its equilibria, that is, the value of $A(t)$ reached upon convergence ($t \to +\infty$). In order to achieve this, I set the right-hand side of Equation 2.21 to zero, substitute B by leveraging Equation 2.20 and solve it for A. This yields two equilibria:

$$A_1 = \frac{-1 + q + z_A + z_B - \sqrt{-4(-1+q)z_A + (-1+q+z_A+z_B)^2}}{2(-1+q)} \tag{2.22}$$

$$A_2 = \frac{-1 + q + z_A + z_B + \sqrt{-4(-1+q)z_A + (-1+q+z_A+z_B)^2}}{2(-1+q)} \tag{2.23}$$

To study the stability of A_1 and A_2, I find the derivative of the right-hand side of Equation 2.21 with respect to A and substitute A with A_1 and A_2. Finally, since the derivative part of the system is composed of one variable and one equation, the derivative with respect to A of the right-hand side of Equation 2.21 corresponds to its eigenvalue, hence the stability conditions are found by checking the values of the parameters q, z_A and z_B for which the obtained expression is lower or equal than zero. From this, A_1 is stable for any reasonable value of the parameters:

$$0 < q \leq 1 \wedge 0 \leq z_A < 1 \wedge 0 \leq z_B < 1 \tag{2.24}$$

These values are in line with the assumptions in Equation 2.7 and the assumption that the amount of zealots doesn't exceed the amount of cooperative agents. In fact, I lock the maximum amount of zealots at half the amount of cooperative agents:

$$z_A + z_B \leq 0.5 \tag{2.25}$$

Figure 2.4: Dynamics of the DSVM system with parameters $z_A = 0$, $q = 0.8$ and $z_B = 0.2$. The arrows show the convergence of $A(t)$ towards the two stable points, colored in black.

This assumption is reasonable as a number of zealots exceeding 50% of the collaborative population would lead to a heavy disruption of the latter's task . Furthermore, it is useful in order to reduce the domain of the parameters. On the other hand, the equilibrium A_2 is unstable for all values of the parameters, except for one particular case:

$$z_A = 0 \land 0 < q \leq 1 \land z_B = 1 - q \tag{2.26}$$

To understand what these two equilibria mean, I substitute the values of the parameters with values that follow the conditions specified in Equation 2.26. For example, they can be set to

$$z_A = 0 \land q = 0.8 \land z_B = 0.2 \tag{2.27}$$

Once I have fixed these parameters, I display the dynamics of Equation 2.21 in Figure 2.4, from which it is clear that A_2 is the stable point corresponding to the outcome in which option B wins, as the system converges towards $A = 0$. Another interesting aspect is that from Figure 2.4 only one equilibrium is visible. The values of A_1 and A_2 with parameters as in Equation 2.27 are:

$$A_1 = 0 \land A_2 = -2.77556 * 10^{-16} \tag{2.28}$$

This leads to the intuition that the two points approximately coincide when the conditions in Equation 2.26 are met and they lead to the victory of option B, meaning that this is a successful wrong addressing attack, since $z_A = 0$. In fact, by substituting the values of parameters z_A and z_B in the equilibria (Equations 2.22 and 2.23) as per the assumption in Equation 2.26:

$$A_1 = A_2 = 0 \tag{2.29}$$

Since A_1 is stable for every reasonable value of the parameters, it follows that its value determines the outcome of the system. This can be seen by choosing some values for the parameters that make A_2 unstable, for example:

$$z_A = 0.1 \land q = 0.8 \land z_B = 0.1 \tag{2.30}$$

Figure 2.5: Dynamics of the DSVM system with parameters $z_A = 0.1$, $q = 0.8$ and $z_B = 0.1$. The arrows show the convergence of $A(t)$ towards the stable point in blue.

Figure 2.6: Dynamics of the DSVM system with parameters $z_A = 0.1$, $q = 0.8$ and $z_B = 0.3$. The arrows show the convergence of $A(t)$ towards the stable point in blue.

The system with parameters set to the values specified in Equation 2.30 evolves as depicted in Figure 2.5, which shows the convergence towards the stable equilibrium at around $A = 0.7$. The unstable one is at around $A = -0.7$, not visible in the domain of A. By increasing the amount of zealots with opinion B and keeping both the quality and the amount of zealots with opinion A as in Equation 2.30, I choose the following parameters in order to show how the alteration of one parameter influences the stable equilibrium:

$$z_A = 0.1 \land q = 0.8 \land z_B = 0.3 \tag{2.31}$$

Figure 2.6 shows the dynamics of the system under these new parameters. The outcome is a deadlock as A converges towards $A = 0.366$. Therefore, by increasing the amount of zealots, the equilibria are shifted towards the negative A axis.
The stable equilibrium is parameterized with respect to z_A, z_B and q, as its value is determined by those quantities. Furthermore, since the stable equilibrium determines the outcome of the system, I define the quorum as the minimum relative amount of agents that need to hold the same opinion in order to consider the agents as having reached consensus. This is needed, as considering a full consensus (100% of the agents holding the same opinion) weakens the fault-tolerance of the system: noise and problems with a high q significantly reduce the likelihood of every single agent agreeing on the best opinion. Therefore, by setting a desired quorum ϵ, I derive the values of the parameters for which the outcome is the victory for option A, option B or results in a deadlock:

$$A_1(z_A, z_B, q) \geq \epsilon \tag{2.32}$$

$$B_1(z_A, z_B, q) \geq \epsilon \implies 1 - A_1(z_A, z_B, q) \geq \epsilon \implies A_1(z_A, z_B, q) \leq 1 - \epsilon \tag{2.33}$$

Inequality 2.32 refers to the conditions on the parameters for which the outcome is the victory of option A, while Inequality 2.33 refers to the victory of B. In the latter, the derivation of B_1 leverages Equation 2.18. The system undergoes a decision deadlock when neither of the two conditions is met:

$$A_1 < \epsilon \vee A_1 > 1 - \epsilon \tag{2.34}$$

The solution of Inequalities 2.32, 2.33 and 2.34 gives rise to three systems of inequalities where z_A, z_B and q can be viewed as variables. The first system refers to the conditions on the parameters for the outcome that favors option A and, by fixing one of the three parameters, I define a functional relationship between the other two parameters. First, I fix $\epsilon = 0.7$ and Inequality 2.32 becomes:

$$\begin{cases} (z_B = 0 \wedge 0 < q < 1) \vee \\ (0 < z_B < 0.3 \wedge 0 < q \leq 1 - 3.33 z_B)\,, & \text{if } z_A = 0 \\ (0 \leq z_B \leq 0.43 z_A \wedge 0 < q < 1) \vee \\ (0.43 z_A < z_B < 0.3 + 0.43 z_A \wedge 0 < q \leq 1 + 1.43 z_A - 3.33 z_B) & \text{if } 0 < z_A < 1 \end{cases} \tag{2.35}$$

The next step is to fix one of the parameters, such as z_A, in order to have a functional relation between the other two parameters. For example, by fixing $z_A = 0.05$, I find the functional relationship $q(z_B)$ by solving the above inequalities. This procedure, repeated for each of the three systems, divides the (z_B, q) plane into three areas, each corresponding to one outcome. In the following, I refer to the (z_B, q) plane as the parameter space. Each model is characterized by its own parameter space that depends on z_A and ϵ. The System 2.35 for the victory of A then becomes:

$$\begin{cases} 0 < q < 1, & \text{if } 0 \leq z_B \leq 0.02, \\ 0 < q \leq 1.07 - 3.33 z_B & \text{if } 0.02 < z_B < 0.32 \end{cases} \tag{2.36}$$

Whereas the condition for the victory of option B becomes:

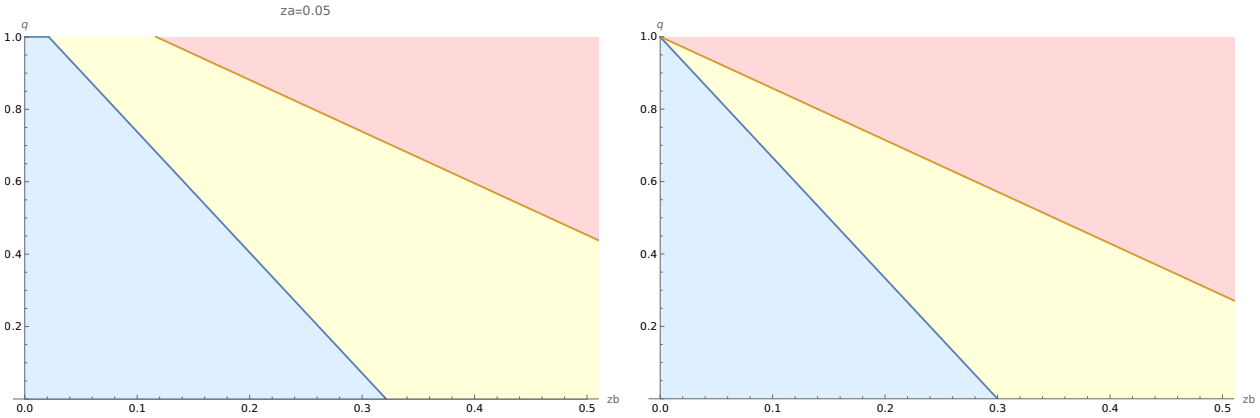
$$1.67 - 1.43 z_B \leq q < 1 \wedge 0.12 < z_B < 0.5 \tag{2.37}$$

Note that z_B is capped at 0.5 in Condition 2.37 as per Assumption 2.25. Then, the lines that divide the parameter space are:

$$f(z_B) = 1.07 - 3.33z_B, \tag{2.38}$$

$$g(z_B) = 1.67 - 1.43z_B, \tag{2.39}$$

The domain of f is $[0.02, 0.32]$, while the domain of g is $[0.12, 0.5]$. The system that leads



(a) Parameters: $\epsilon = 0.7$, $z_A = 0.05$ (b) Parameters: $\epsilon = 0.7$, $z_A = 0$

Figure 2.7: The DSVM parameter space divided in three regions: the blue region corresponds to option A winning, the yellow region corresponds to a decision deadlock and the red region corresponds to option B winning.

to a decision deadlock fills up the rest of the parameter space, as can be seen in Figure 2.7a. The wrong addressing attack, with $z_A = 0$, leads to the definition of the parameter space as in Figure 2.7b. The latter has been obtained by setting $z_A = 0$ in Condition 2.35. Then, I use the same procedure to obtain the inequalities that lead to the victory of option B. Finally, I infer the decision deadlock space as the region in which neither of the other two outcomes lie. The parameter space of Figure 2.7b is divided by the following functions:

$$f(z_B) = 1 - 3.33z_B \tag{2.40a}$$

$$g(z_B) = 1 - 1.43z_B \tag{2.40b}$$

By comparing Equations 2.38, 2.39 and Equations 2.40a, 2.40b I note that the slope is the same, a fact that I also observe in the structure of Inequalities 2.35, where the addends dependant on z_A do not multiply z_B, thus they only concur to the alteration of the intercept of the lines that divide the parameter space. Therefore, the z_A parameter shifts the lines along the q axis: the higher z_A, the higher the intercept. In particular, by observing Figures 2.7a and 2.7b, following the decrease of z_A, I note an increase in

the total area corresponding to the victory of option B, as well as a decrease in the area corresponding to the victory of option A. This is due to the shift of the lines that delimit the regions of the parameter space, which also causes an decrease in the deadlock region. In particular, the introduction of more zealots with opinion A results in an increase of the yellow area: this sheds light on the fact that under a wrong addressing attack, the DSVM model is more incline to converging to option B, whereas under a denial of service attack, the model undergoes a decision deadlock more often.

2.3.2. Cross-Inhibition with voter model

The cross-inhibition voter model (**CIVM** from now on) leverages Equations 2.19 and 2.13. From the former, the relative number of agents in the uncommitted state at a given time is:

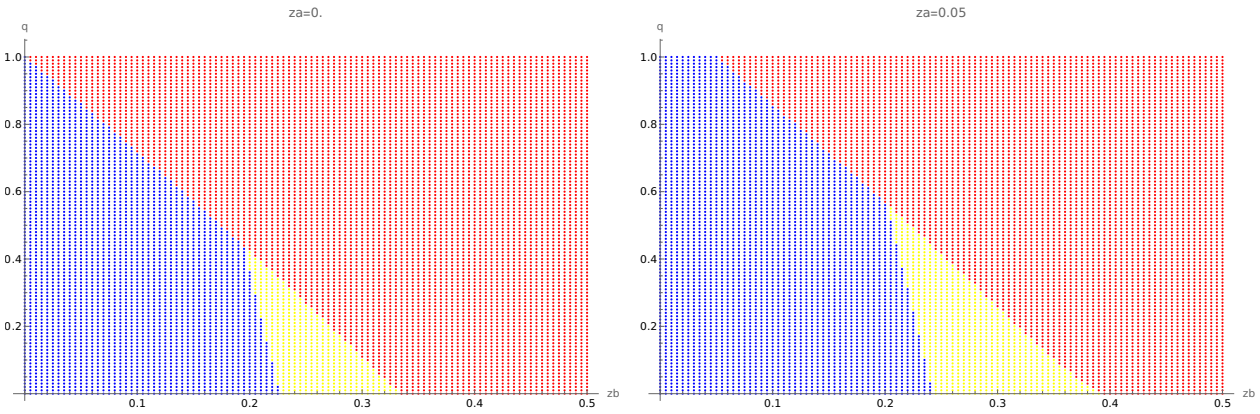
$$U(t) = 1 - A(t) - B(t) \tag{2.41}$$

In order to define the system as a whole, I need two more equations that model the change over time in the relative populations of agents with opinion A and B:

$$\begin{cases} \frac{dA}{dt}(t) = P_1\left(n_A^{\#}(t)\right)U(t) - P_1\left(n_B^{\#}(t)\right)A(t) = n_A^{\#}(t)U(t) - n_B^{\#}(t)A(t) & \text{(2.42a)} \\ \frac{dB}{dt}(t) = P_1\left(n_B^{\#}(t)\right)U(t) - P_1\left(n_A^{\#}(t)\right)A(t) = n_B^{\#}(t)U(t) - n_A^{\#}(t)B(t) & \text{(2.42b)} \end{cases}$$

The first term of Equation 2.42a refers to the increase in the A population whenever an agent in the uncommitted state receives a message containing an A opinion: this is given by the product of the probability of receiving an A message, multiplied by the relative amount of agents in the uncommitted state; whereas the second term refers to the decrease in the A population whenever an agent with opinion A receives a message with opinion B. The case for Equation 2.42b is symmetric, as the first term refers to the increase in the B population whenever an agent in the uncommitted state receives a message containing opinion B, whereas the second term refers to the decrease in the B population whenever an agent holding opinion B receives a message containing opinion A. The equilibria of this model are not calculated in a parametric fashion, rather numerically: by exploring the parameter space, with some initial conditions, the system is integrated over time for a large time interval and then the final value of populations A and B are checked to determine the outcome of the system.

An example of the generated parameter space can be seen in Figure 2.8a, where the system has been integrated over time for $t_{MAX} = 5*10^8$ for each point. The points have been chosen starting from $(0, 0)$ until $(1, 0.5)$, with a resolution of 1% and by setting $z_A = 0$,



(a) $z_A = 0$ (b) $z_A = 0.05$

Figure 2.8: The CIVM parameter space divided in two regions: the blue region corresponds to option A winning the red region corresponds to option B winning. Common parameters: $A(0) = 0.5 + 10^{-6}$, $t_{MAX} = 5 * 10^8$, $\epsilon = 0.7$.

making this the parameter space of CIVM in the case of a wrong addressing attack. By comparing CIVM (Figure 2.8a) with DSVM (Figure 2.7b), it is possible to state that the former is less accurate, as the red region is bigger; nevertheless CIVM is able to break deadlocks more often, as can be seen by the greatly reduced yellow region.

The ability of cross-inhibition to break deadlocks more often than the direct switch update rule is a known result, as described in [13].

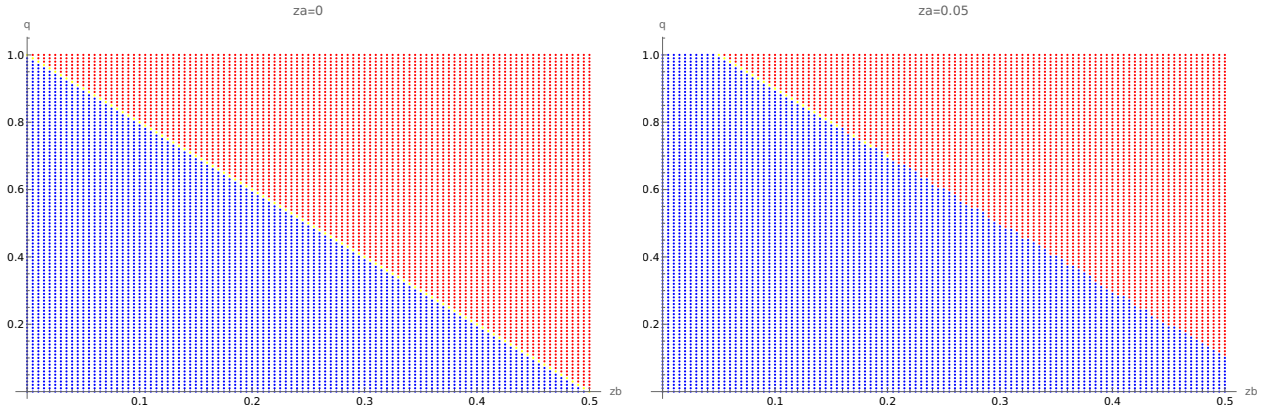
2.3.3. Direct switch with majority rule

I implement the direct switch with majority rule (**DSMR**) model by leveraging Equation 2.12 and Equation 2.18. By following the approach used in the development of the other homogeneous models, the following system fully describes the model:

$$\begin{cases} B(t) = 1 - A(t) \\ \dfrac{dA}{dt}(t) = P_0\left(n_A^{\#}(t)\right) B(t) - P_0\left(n_B^{\#}(t)\right) A(t) \end{cases} \tag{2.43}$$

Where P_0 is approximated as in Approximation 2.14. The first term in the derivative of A denotes an increase in the A population given by the probability that more than half of the messages received by a B agent contain option A; similarly, the second term denotes a decrease in the A population whenever more than half of the messages an A agent receives

contain option B. For convenience, in the following, I will refer to this model as P_α-DSMR.



(a) $z_A = 0$ (b) $z_A = 0.05$

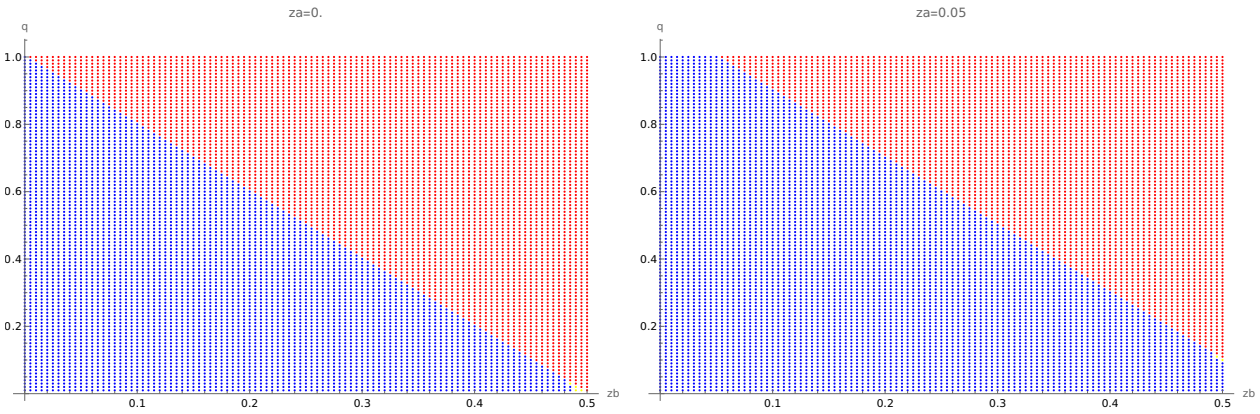
Figure 2.9: The P_α-DSMR parameter space divided in three regions: the blue region corresponds to option A winning, the red region corresponds to option B winning and the yellow region corresponds to a decision deadlock. Common parameters: $A(0) = 0.5+10^{-6}$, $t_{MAX} = 5 * 10^8$, $\epsilon = 0.7$.

By comparing Figures 2.9a and 2.9b and by considering the accuracy and the regret metric that will be described in Section 2.4, I argue that the increase in zealots with opinion A allows for a greater total accuracy of the model and a decrease in the total regret, as can be seen from the increase of the total blue area and the decrease of the yellow area. Furthermore, by comparing them to Figures 2.7b and 2.7a, the majority rule shows to be superior both for the blue area being bigger in the P_α-DSMR and the yellow area being smaller.

On the other hand, the model that leverages Equation 2.15 is built as follows:

$$\begin{cases} B(t) = 1 - A(t) \\ \dfrac{dA}{dt}(t) = p_A(t)B(t) - p_B(t)A(t) \end{cases} \tag{2.44}$$

The structure of the two models is the same, what changes is how the probability of receiving a message is calculated, which is discussed in Section 2.1. For convenience, in the following, I refer to this model as C-DSMR (Combinatorial-DSMR).



(a) $z_A = 0$ (b) $z_A = 0.05$

Figure 2.10: The C-DSMR parameter space divided in two regions: the blue region corresponds to option A winning the red region corresponds to option B winning. Common parameters: $A(0) = 0.5 + 10^{-6}$, $t_{MAX} = 5 * 10^8$, $\epsilon = 0.7$, G=4.

Finally, by analyzing Figures 2.10a and 2.10b, the blue area in the C-DSMR model is the same as in P_α-DSMR model, with the former suffering from deadlocks around the line $q = 1 - 2z_B$, while the latter suffers from sporadic decision deadlocks for high values of z_B and low values of q.

I conclude that the majority rule is both better in terms of accuracy and in terms of how often it breaks deadlocks, when using direct switch.

2.3.4. Cross-inhibition with majority rule

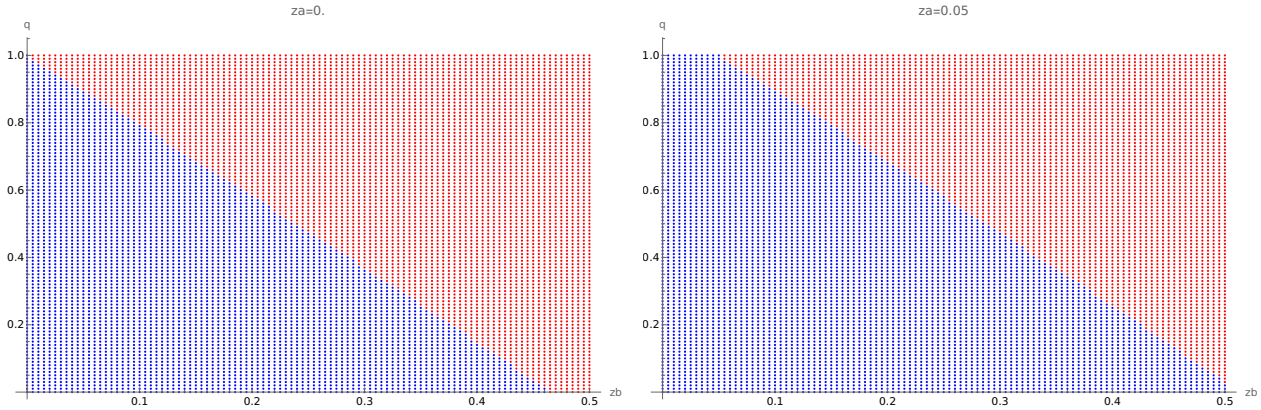
I develop the cross-inhibition with majority rule (**CIMR**) model both with the P_α probability function (Equation 2.12) and with the combinatorial probability function (Equations 2.15 and 2.16). Both approaches leverage Equation 2.19. I build the former as follows:

$$\begin{cases} U(t) = 1 - A(t) - B(t) \\ \dfrac{dA}{dt}(t) = P_0\left(n_A^{\#}(t)\right) U(t) - P_0\left(n_B^{\#}(t)\right) A(t) \\ \dfrac{dB}{dt}(t) = P_0\left(n_B^{\#}(t)\right) U(t) - P_0\left(n_A^{\#}(t)\right) B(t) \end{cases} \tag{2.45}$$

The first equation derives the relative number of agents in the uncommitted state, whereas the second and third equations describe the evolution of the populations of committed agents. In the latter, the first term is related to the increase of the committed population

i given by the probability that an uncommitted agent receives opinion i, while the second term is related to the decrease of the committed population i given by the probability that a committed agent i becomes uncommitted, where i=A,B. In the following, I refer to this model as P_α-CIMR.

The parameter space of the P_α-CIMR model in Figures 2.11a and 2.11b shows that the model suffers from no deadlocks, as the yellow region is absent. By comparing it to the CIVM model (Figures 2.8a, 2.8b), the total blue area is larger, which shows that the majority rule has a tendency to be more accurate with respect to the voter model. Hence, under a wrong addressing attack and with cross-inhibition agents, the majority rule is both more accurate and it breaks decision deadlocks more often than the voter model. Furthermore, by comparing it with the P_α-DSMR model (Figures 2.9a, 2.9b), the parameter space is roughly split in half, but the P_α-DSMR model still suffers from some deadlocks, which are absent in the P_α-CIMR model, while the latter has a lower blue region. The same trend is present when comparing the P_α-CIMR model with the C-DSMR model, however, in the latter the deadlock area is negligible compared with the decrease in the blue area.



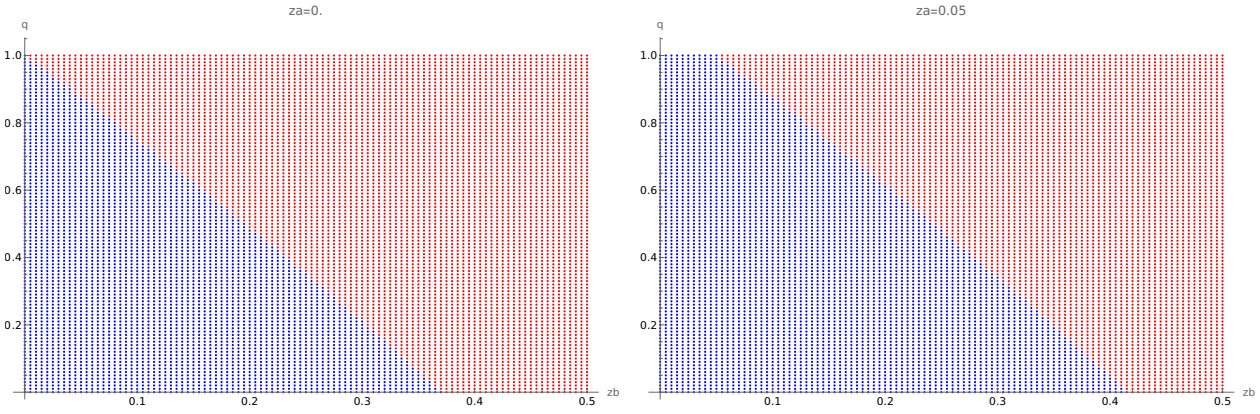
(a) $z_A = 0$ (b) $z_A = 0.05$

Figure 2.11: The P_α-CIMR parameter space divided in two regions: the blue region corresponds to option A winning the red region corresponds to option B winning. Common parameters: $A(0) = 0.5 + 10^{-6}$, $t_{MAX} = 5 * 10^8$, $\epsilon = 0.7$.

On the other hand, I build the model that leverages Equations 2.15 and 2.16, C-CIMR (Combinatorial-CIMR) from now on, as follows:

$$\begin{cases} U(t) = 1 - A(t) - B(t) \\ \dfrac{dA}{dt}(t) = p_A(t)U(t) - p_B(t)A(t) \\ \dfrac{dB}{dt}(t) = p_B(t)U(t) - p_A(t)B(t) \end{cases} \tag{2.46}$$

As in the case of DSMR, the structure of the P_α-CIMR model and of the C-CIMR model is the same, but the probability of receiving a message with a specific option is calculated differently. In the former, I use Equation 2.14, while the latter leverages Equations 2.15 and 2.16.



(a) $z_A = 0$ (b) $z_A = 0.05$

Figure 2.12: The C-CIMR parameter space divided in two regions: the blue region corresponds to option A winning the red region corresponds to option B winning. Common parameters: $A(0) = 0.5 + 10^{-6}$, $t_{MAX} = 5 * 10^8$, $\epsilon = 0.7$, G=4.

Figures 2.12a, 2.12b show the parameter space of the C-CIMR model for $z_A = 0$ and $z_A = 0.05$, respectively. By comparing it to the P_α-CIMR parameter space in Figures 2.11a, 2.11b, the blue region is smaller due to the fact that the line separating the blue and the red regions has a steeper slope. However, the comparison done with respect to the other models is the same as for the P_α-CIMR model, with the only distinction being the smaller blue area.

I conclude that the models using the majority rule have a better performance with respect to the voter model both in terms of accuracy and of how often they break deadlocks. This property is present both for the models that use the direct switch and those that use the cross-inhibition update rules.

2.3.5. Heterogeneous direct switch

In this section, I develop the models where all the agents use the direct switch update rule and a percentage of the population uses the voter model, while the rest use the majority rule. Therefore, I introduce a parameter k to denote the percentage of agents that use the voter model filtering mechanism. It follows that in the homogeneous voter models k is equal to 1, whereas in the homogeneous majority rule models k is set to 0. This calls for the introduction of two variables: one to keep track of the relative number of agents holding opinion A at time t that are using the voter model and the other to keep track of the relative number of agents holding opinion A at time t that are using the majority rule. I apply the same logic to agents holding opinion B at time t, introducing a variable to keep track of agents holding opinion B and using, respectively, the voter model and the majority rule. However, the relative number of agents holding B is defined in Equation 2.18, thus it is sufficient to define the variation in the two populations of agents that hold opinion A. Let $A_{VM}(t)$ (respectively, $B_{VM}(t)$) be the relative number of agents holding opinion A (B) at time t while using the voter model and let the $A_{MR}(t)$ ($B_{MR}(t)$) be the relative number of agents holding opinion A (B) at time t while using the majority rule. Then:

$$\begin{aligned} A_{VM}(t) + B_{VM}(t) = k &\implies B_{VM}(t) = k - A_{VM}(t) \\ A_{MR}(t) + B_{MR}(t) = 1 - k &\implies B_{MR}(t) = 1 - k - A_{MR}(t) \end{aligned}$$

These two new populations call for the redefinition of $n_A^{\#}(t)$ and of $n_B^{\#}(t)$:

$$\begin{aligned} n_A^{\#}(t) &= \frac{A_{VM}(t) + A_{MR}(t) + z_A}{A_{VM}(t) + A_{MR}(t) + z_A + q(B_{VM}(t) + B_{MR}(t)) + z_B} \\ n_B^{\#}(t) &= \frac{B_{VM}(t) + B_{MR}(t) + z_B}{A_{VM}(t) + A_{MR}(t) + z_A + q(B_{VM}(t) + B_{MR}(t)) + z_B} \end{aligned}$$

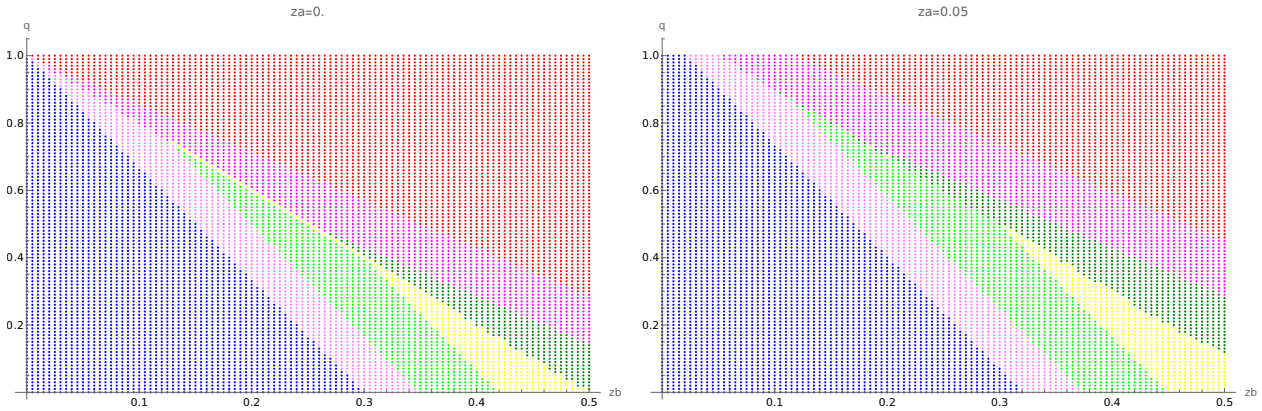
The next step is defining the change in each A population, which can be done leveraging the reasoning behind Equation 2.21 together with the approximation in Equation 2.14:

$$\begin{cases} \dfrac{dA_{VM}}{dt}(t) = n_A^{\#}(t)B_{VM}(t) - n_B^{\#}(t)A_{VM}(t) & (2.47) \\ \dfrac{dA_{MR}}{dt}(t) = P_0(n_A^{\#}(t))B_{MR}(t) - P_0(n_B^{\#}(t))A_{MR}(t) & (2.48) \end{cases}$$

Equation 2.47 models the change in the A population that uses the voter model and has the same structure as the homogeneous models: the first term refers to the increase of the population whenever an agent holding opinion B and using the voter model receives a message containing opinion A; the second term refers to the decrease of the population

whenever an agent holding opinion A receives a message containing opinion B. Equation 2.48 models the change in the A population that uses the majority rule and it has the same structure as Equation 2.47. Notice that the message may come from an agent using either voting mechanism, but solely the specific receiving agent's voting mechanism is taken into account in each equation. Then, Equations 2.48 and 2.47 constitute the P_α heterogeneous direct switch model, or P_α-HDS. The same structure, keeping in mind the redefinition of $n_A^\#(t)$ and of $n_B^\#(t)$ to be used inside of Equations 2.15 and 2.16, may be used to define the combinatorial heterogeneous direct switch model (C-HDS):

$$\begin{cases} \dfrac{dA_{VM}}{dt}(t) = n_A^\#(t)B_{VM}(t) - n_B^\#(t)A_{VM}(t) & (2.49) \\ \dfrac{dA_{MR}}{dt}(t) = p_A(t)B_{MR}(t) - p_B(t)A_{MR}(t) & (2.50) \end{cases}$$

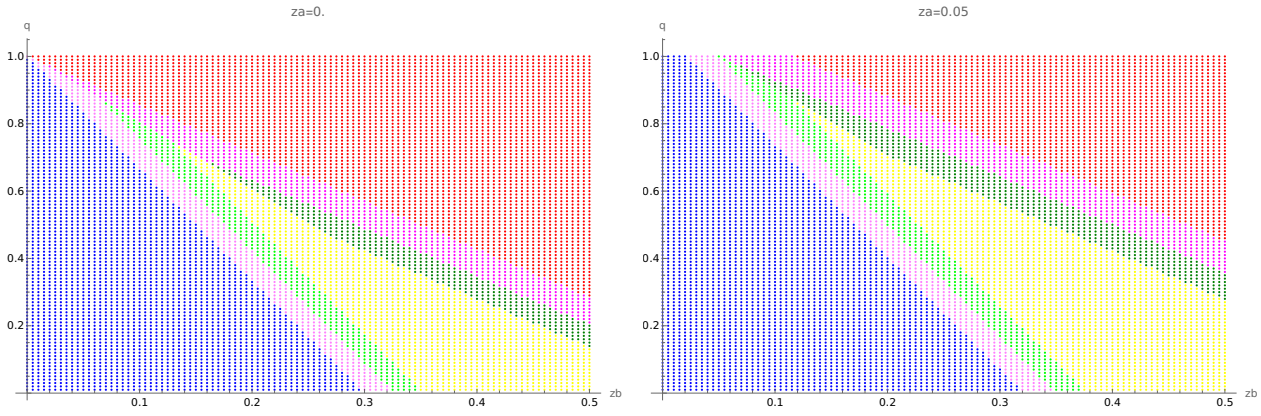


(a) $z_A = 0$ (b) $z_A = 0.05$

Figure 2.13: The P_α-HDS parameter space divided in seven regions: the blue region corresponds to option A winning for $k \in [0.8, 1]$, the red region corresponds to option B winning for $k \in [0.8, 1]$, the yellow region corresponds to a decision deadlock for $k \in [0.8, 1]$, the pink region corresponds to option A winning for $k = [0.8, 0.9]$ and to a decision deadlock for $k = 1$, the light green region corresponds to option A winning for $k = 0.8$ and to a decision deadlock for $k = [0.9, 1]$, the dark green region corresponds to option B winning for $k = [0.8, 0.9]$ and to a decision deadlock for $k = 1$, the magenta region corresponds to option B winning for $k = [0.8, 0.9]$ and to a decision deadlock for $k = 1$. Common parameters: $A(0) = 0.5 + 10^{-6}$, $t_{MAX} = 5 * 10^8$, $\epsilon = 0.7$.

The evolution of the parameter space of the P_α-HDS model for $k = 0.8, 0.9, 1$ is shown in Figures 2.13a and 2.13b with $z_A = 0$ and $z_A = 0.05$ respectively. The plots show the decrease of the deadlock area as k decreases. In fact, the deadlock area common to all $k \in [0.8, 1]$ corresponds to the yellow area: this is the region where all the models

undergo a decision deadlock. Furthermore, when $k = 0.8$, the light green and the pink areas correspond to option A winning, together with the blue area. On the other hand, the dark green, the magenta and the red areas correspond to option B winning when $k = 0.8$. By increasing the value of k to 0.9, the light green area becomes a deadlock region, together with the dark green area. Moreover, by increasing k to 1, the pink and the magenta areas become deadlock regions as well.



(a) $z_A = 0$ (b) $z_A = 0.05$

Figure 2.14: The C-HDS parameter space divided in seven regions: the blue region corresponds to option A winning for $k \in [0.8, 1]$, the red region corresponds to option B winning for $k \in [0.8, 1]$, the yellow region corresponds to a decision deadlock for $k \in [0.8, 1]$, the pink region corresponds to option A winning for $k = [0.8, 0.9]$ and to a decision deadlock for $k = 1$, the light green region corresponds to option A winning for $k = 0.8$ and to a decision deadlock for $k = [0.9, 1]$, the dark green region corresponds to option B winning for $k = [0.8, 0.9]$ and to a decision deadlock for $k = 1$, the magenta region corresponds to option B winning for $k = [0.8, 0.9]$ and to a decision deadlock for $k = 1$. Common parameters: $A(0) = 0.5 + 10^{-6}$, $t_{MAX} = 5 * 10^8$, $\epsilon = 0.7$, $G = 4$.

Figures 2.14a, 2.14b show the parameter space of the C-HDS model for $k = 0.8, 0.9, 1$ and $z_A = 0$, $z_A = 0.05$, following the same rationale as Figures 2.13a, 2.13b.

2.3.6. Heterogeneous cross-inhibition

The development of the heterogeneous cross-inhibition models follows the same reasoning as the heterogeneous direct switch models, with the addition of a variable to keep track of the relative number of agents in the uncommitted state that use a specific voting mechanism. As before, let k be the percentage of agents using the voter model and let $U_{VM}(t)$ and $U_{MR}(t)$ be the relative number of agents in a latent state that use the voter

model and the majority rule, respectively. Then:

$$U_{VM}(t) = k - A_{VM}(t) - B_{VM}(t) \tag{2.51}$$
$$U_{MR}(t) = 1 - k - A_{MR}(t) - B_{MR}(t) \tag{2.52}$$

Following the reasoning behind the CIVM and the CIMR models (Equations 2.42a,2.42b and 2.45) and using the P_0 approximation in Equation 2.14, the following P_α heterogeneous cross-inhibition model (P_α-HCI from now on) may be constructed:

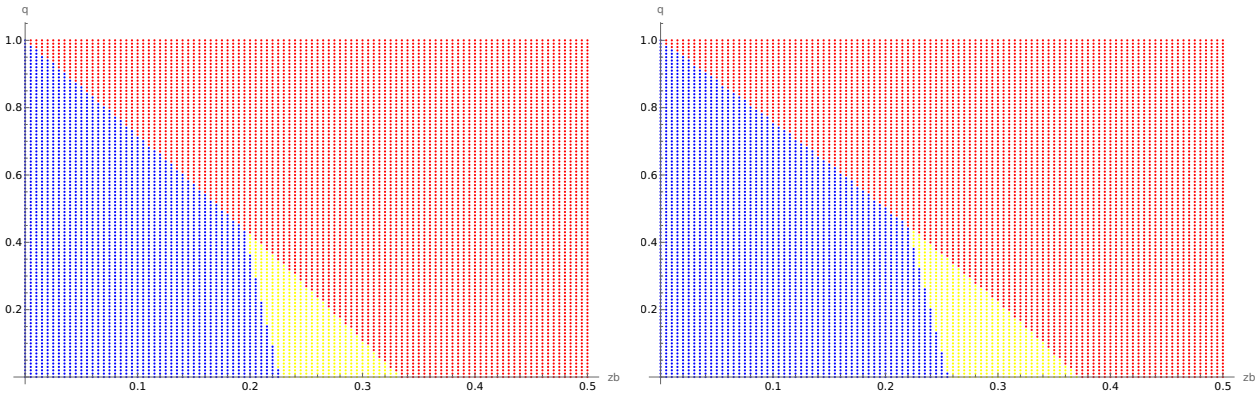
$$\begin{cases} \frac{dA_{VM}}{dt}(t) = n_A^{\#}(t)U_{VM}(t) - n_B^{\#}(t)A_{VM}(t) & (2.53) \\ \frac{dB_{VM}}{dt}(t) = n_B^{\#}(t)U_{VM}(t) - n_A^{\#}(t)B_{VM}(t) & (2.54) \\ \frac{dA_{MR}}{dt}(t) = P_0\left(n_A^{\#}(t)\right)U_{MR}(t) - P_0\left(n_B^{\#}(t)\right)A_{MR}(t) & (2.55) \\ \frac{dB_{MR}}{dt}(t) = P_0\left(n_B^{\#}(t)\right)U_{MR}(t) - P_0\left(n_A^{\#}(t)\right)B_{MR}(t) & (2.56) \end{cases}$$

Equations 2.53, 2.54 have the same structure as the CIVM model, with the peculiarity that the agents that receive the message are part of the voter model sub-population, whereas Equations 2.55, 2.56 have the same structure as the P_α-CIMR model. Modifying this model to derive the combinatorial form is straightforward, as only the probability function in the majority rule population needs to be changed:

$$\begin{cases} \frac{dA_{VM}}{dt}(t) = n_A^{\#}(t)U_{VM}(t) - n_B^{\#}(t)A_{VM}(t) \\ \frac{dB_{VM}}{dt}(t) = n_B^{\#}(t)U_{VM}(t) - n_A^{\#}(t)B_{VM}(t) \\ \frac{dA_{MR}}{dt}(t) = p_A(t)U_{MR}(t) - p_B(t)A_{MR}(t) \\ \frac{dB_{MR}}{dt}(t) = p_B(t)U_{MR}(t) - p_A(t)B_{MR}(t) \end{cases} \tag{2.57}$$

Therefore, the model in Equation 2.57 denotes the combinatorial heterogeneous cross-inhibition model, or C-HCI from now on.

The evolution of the parameter space of the P_α-HCI model is shown for the values of $k = 1$ and $k = 0.9$ in Figures 2.15a and 2.15b for $z_A = 0$. In this model, the parameter space shifts with respect to the z_B axis, making it impossible to show a plot as in the case of the P_α-HDS and of the C-HDS models, since there is an overlap across the yellow and blue regions for decreasing values of k.



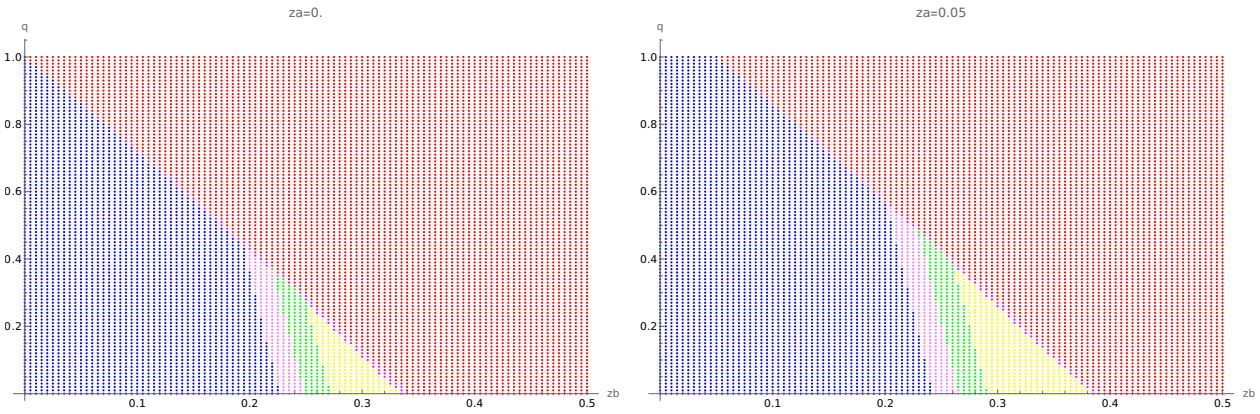
(a) $k = 1$ (b) $k = 0.9$

Figure 2.15: The P_α-HCI parameter space divided in three regions: the blue region corresponds to option A winning, the red option corresponds to option B winning and the yellow region corresponds to a decision deadlock. Common parameters: $A(0) = 0.5+10^{-6}$, $t_{MAX} = 5 * 10^8$, $\epsilon = 0.7$, $z_A = 0$.

On the other hand, I plot the evolution of the parameter space of the C-HCI in Figures 2.16a and 2.16b for $z_A = 0$ and $z_A = 0.05$, respectively. The plots show the decrease of the deadlock area as k decreases. In particular, the common deadlock area for all $k \in [0.8, 1]$ corresponds to the yellow area: in this region, all the three models undergo a decision deadlock. Furthermore, when $k = 0.8$, the light green area and the pink area correspond to option A winning, together with the blue area. On the other hand, the magenta and the red areas correspond to option B winning when $k = 0.8$. By increasing the value of k to 0.9, the light green area becomes a deadlock region. Moreover, by increasing k to 1, the pink and the magenta areas become deadlock regions as well. Comparing these figures to Figures 2.14a, 2.14b the deadlock area that turns to a blue region is smaller. This is due to the ability of cross-inhibition to break deadlocks more easily than the direct switch update rule, thus the evolution of the parameter space in terms of decrease of the deadlock area is less advantageous. Moreover, in Figures 2.16a and 2.16b the dark green area is missing, meaning that by increasing the value of k from 0.9 to 1 the deadlock area does not become a red region.

2.4. Metrics

The evaluation of mathematical models leverages the use of some metrics in order to compare and rank them. I propose the following metrics: accuracy, regret, convergence time and cognitive cost. Accuracy refers to the number of points in the parameter space



(a) $z_A = 0$ (b) $z_A = 0.05$

Figure 2.16: The C-HCI parameter space divided in six regions: the blue region corresponds to option A winning for $k \in [0.8, 1]$, the red region corresponds to option B winning for $k \in [0.8, 1]$, the yellow region corresponds to a decision deadlock for $k \in [0.8, 1]$, the pink region corresponds to option A winning for $k = [0.8, 0.9]$ and to a decision deadlock for $k = 1$, the light green region corresponds to option A winning for $k = 0.8$ and to a decision deadlock for $k = [0.9, 1]$, the magenta region corresponds to option B winning for $k = [0.8, 0.9]$ and to a decision deadlock for $k = 1$. Common parameters: $A(0) = 0.5 + 10^{-6}$, $t_{MAX} = 5 * 10^8$, $\epsilon = 0.7$, $G = 4$.

for which a model converges to some $A(t_{MAX}) \geq \epsilon$, over the total number of points in the discretized parameter space. On the other hand, regret allows to give a weight to each point in the parameter space in such a way that the regret of a point for which the system converges to A is equal to zero, while the regret of a point for which the system converges to B is the difference in the qualities, whereas the regret of a point for which the system undergoes a decision deadlock is equal to the quality of option A. This metric punishes models that suffer from decision deadlocks, such as the models using the direct switch opinion update rule. Moreover, the convergence time refers to the average minimum t^* such that $A(t^*) \geq \epsilon$; therefore, only the points for which the outcome of the system is option A are considered. Finally, the cognitive cost is a metric that considers the upper bound of the total number of messages that are processed in a model, punishing models that need more messages than others, such as the ones using the majority rule.

2.4.1. Accuracy

The accuracy of a model refers to the number of points which result in the victory of option A, over the total number of points in the parameter space. It is a percentage, ranging from 0 to 1, where 0 means that no point in the parameter space allows for the

convergence towards A and 1 means that every point leads to the convergence to A. Let the accuracy of a point in the parameter space be defined as:

$$\text{Accuracy}(z_B, q) = \begin{cases} 1 & \text{if } A(z_B, q, t_{MAX}) \geq \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{2.58}$$

The accuracy of a model is then calculated as the total area of the parameter space corresponding to the convergence of the system to option A, over the total area of the parameter space. Let Δ denote the resolution used to discretize uniformely the parameter space. Then, the total area of the parameter space is equal to 0.5Δ, as the number of zealots B is limited to 0.5 and the quality lies in the range $[0, 1]$. Therefore:

$$\text{Accuracy} = \frac{\sum_{z_B=0}^{0.5} \sum_{q=0}^{1} Accuracy(z_B, q)}{0.5\Delta} \tag{2.59}$$

The objective of a model is to maximize this metric, therefore the higher the accuracy of a model, the better.

2.4.2. Regret

The regret of a model represents the total value that has been lost due to decision deadlocks and the convergence of the system to option B. The value, in this case, corresponds to the quality of the option to which the system has converged. In order to represent the regret of a model, first it is necessary to define the regret of a single point inside of the parameter space:

$$Regret(z_B, q) = \begin{cases} 0, & \text{if } A(z_B, q, t_{MAX}) \geq \epsilon \\ 1 - q_B, & \text{if } A(z_B, q, t_{MAX}) \leq 1 - \epsilon \\ 1 & \text{otherwise} \end{cases} \tag{2.60}$$

The rationale behind Equation 2.60 is the following: if the agents converge towards the option with maximum quality, then they shall regret nothing, since they have chosen the best option; on the other hand, if they converge towards the worst option, they regret the difference between the option they have chosen and the best option; otherwise, if they undergo a decision deadlock, the total value they have lost is equal to the maximum quality. Finally, it is possible to compute the total regret of a model as the summation

of the regrets over the parameter space:

$$\text{Regret} = \sum_{z_B=0}^{0.5} \sum_{q=0}^{1} Regret(z_B, q) \tag{2.61}$$

The objective of a model is to minimize this metric, thus the lower the regret of a model, the better.

2.4.3. Convergence time

The convergence time is the average number of integrations needed for the system of ODEs to converge to option A. The concept of integration has been introduced in Section 2.3.2 in order to find the equilibria of complex models. The calculation of the average convergence time is comprised solely of configurations for which the system converges to A: whenever the system converges to B or to a deadlock, the time is not taken into account. Let $\{A(z_B, q, t_i)\}_{i=0}^{t_{MAX}}$ be the series of values taken by $A(z_B, q, t_i)$ at the i-th integration; then the convergence time of a point in the parameter space is calculated as follows:

$$ConvergenceTime(z_B, q) = \begin{cases} \min_{t^*} A(t^*, z_B, q) \geq \epsilon & \text{if } A(t_{MAX}, z_B, q) \geq \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{2.62}$$

Finally, the average convergence time of a model may be calculated as the cumulative sum of the convergence time across the parameter space, divided by the number of points that contributed to the cumulative sum. The latter coincides with the accuracy multiplied by the total number of points in the discretized parameter space, as only the points corresponding to the convergence of the system to option A are considered in the average:

$$\sum \text{Convergence Time} = \sum_{z_B=0}^{0.5} \sum_{q=0}^{1} ConvergenceTime(z_B, q)$$

$$\text{Convergence Time} = \frac{\sum \text{Convergence Time}}{0.5\Delta\text{Accuracy}} \tag{2.63}$$

Moreover, the standard deviation of a model may be calculated by the following formula:

$$\text{Standard Deviation} = \sqrt{\frac{\sum_{z_B=0}^{0.5} \sum_{q=0} \left[\text{Convergence Time}(z_B, q) - \text{Convergence Time}\right]^2}{0.5\Delta\text{Accuracy} - 1}} \tag{2.64}$$

2.4.4. Cognitive cost

The cognitive cost of a model is defined as the maximum number of messages processed by that model. To introduce this metric, first I compute the number of messages processed by a single agent. At each time step, in the voter model, each agent has to process 1 message; on the other hand, in the majority rule, each agent has to process G messages, where G is the average group size, which corresponds to the number of messages an agent receives in average. Then, I take a pessimistic approach towards how many rounds of voting are needed and define it as the sum of the average convergence time of the model and its standard deviation. Finally, the total cost will be given by multiplying the linear combination of the number of messages processed by the voter model and the majority rule based on the k parameter, by the upper bound on the number of rounds of voting and by the total number of agents:

$$c = Nt\left[k + (1-k)\,G\right] \tag{2.65}$$

Equation 2.65 is composed as follows:

- $k + (1-k)\,G$: k is the percentage of agents that use the voter model. Each agent processes either 1 message per round, if it uses the voter model, or G messages per round, if it uses the majority rule;
- Nt refers to the total number of agents times the average convergence time of the model. This defines the upper bound on the number of rounds of voting, times the total number of agents.

3 | Results

In this chapter, I analyze the accuracy, the regret and the convergence time of the heterogeneous models, namely of P_α-HDS, C-HDS, P_α-HCI and C-HCI, across different percentages of agents using the voter model opinion filtering mechanism, thus I vary the k parameter. Specifically, I realize the analysis of each model by discretizing the k parameter and fixing it to a value in the range $[0, 1]$. In this range, I use a variation of 10% in order to have a parameter space, under a specific value of z_A, from which I extract the metrics described in Section 2.4. Furthermore, I compare the accuracy and cognitive cost when modulating the percentage of agents that use the voter model. In order to assess the value of k that assures the best trade-off between accuracy and cognitive cost in a model, I normalize the latter with respect to the former, thus transforming each cost by multiplying it by the following factor:

$$\chi = \frac{a_{MAX}}{c_{MAX}} \tag{3.1}$$

where a_{MAX} and c_{MAX} are the maximum accuracy and the maximum cost, respectively, of a model across the values of k. The logic behind this is that the model having maximum accuracy has also maximum cognitive cost and, following the analysis done on the homogeneous models in Section 2, that is reflected by models using the majority rule, rather than the voter model, hence for $k = 0$. Thus, the multiplication with χ transforms the coordinates of the accuracy versus cognitive cost graph in such a way that the distances become comparable, since they are normalized one with respect to the other. Finally, the best trade-off is reached when the Euclidean distance between the occupied point of a model in the normalized accuracy versus cognitive cost graph and the point $(0, a_{MAX})$ is minimum. In mathematical form:

$$k^* = \left\{ k \in [0, 1] | \min_k \sqrt{(\chi c_k)^2 + (a_{MAX} - a_k)^2} \right\} \tag{3.2}$$

In Equation 3.2, c_k and a_k refer to the cognitive cost and to the accuracy, respectively, of the heterogeneous model having that particular k value.

Moreover, I use two different models for each update opinion rule, each based on a particular definition of the probability of receiving a message, in order to validate the results I have found in the analysis, since it shows that despite the modelling behind it, the overall behaviour is common across the two approaches.
To take into account different possible initial conditions, when analyzing heterogeneous models, the metrics I show are averaged out on three possible initial conditions, namely: $A(0) = 0.5 + 10^{-6}$, which corresponds to a slight advantage for the A population, $A(0) = B(0) = 0.5$, which corresponds to equal initial populations, and $A(0) = 0.5 - 10^{-6}$, which corresponds to a slight advantage for the B population. Furthermore, for each model, in order to assess the resilience of the swarm, I analyze two different cases:

- $z_A = 0$, which corresponds to the swarm being subject to a wrong addressing attack;
- $z_A = 0.05$, which includes the denial of service attack, since this attack defines a functional relationship between the number of zealots and the quality of the worst option; thus it corresponds to a function $q(z_B)$, which is lies in the parameter space.

I obtain the following results through the Wolfram Mathematica software and the relative code can be found in Appendix A.

3.1. P_α-HDS

I plot the accuracy and regret of the P_α-HDS model with respect to k, with $z_A = 0$ (Figure 3.1) and with $z_A = 0.05$ (Figure 3.2). The accuracy of the P_α-HDS model with $z_A = 0$ decreases as the percentage of voter model agents increases. In particular, the maximum accuracy is around 0.49 when all the agents are using the majority rule ($k = 0$), while the minimum is around 0.3 when all the agents are using the voter model. On the other hand, the regret follows the inverse trend, having the lowest value when using the majority rule and the highest value when using the voter model. Moreover, I note that the trend is not linear, in particular, the addition of small percentages of majority rule agents increase the accuracy and decreases the regret, from $k = 0.9$ until $k = 0.7$, where the benefits cease. The same trend can be observed in Figure 3.2, where the amount of zealots A is fixed to $z_A = 0.05$. Here, the main difference is in terms of the maximum and minimum accuracy achieved: both have higher values with respect to the case of $z_A = 0$, a natural consequence of the increment of zealots with opinion A, which boosts the overall accuracy. However, the interesting difference is in relation with the regret, which has a lower value in the case of $k = 0$, but a higher value in the case of $k = 1$, meaning that in terms of this metric, the voter model is more susceptible to a denial of service attack, whereas the majority rule is more resilient.

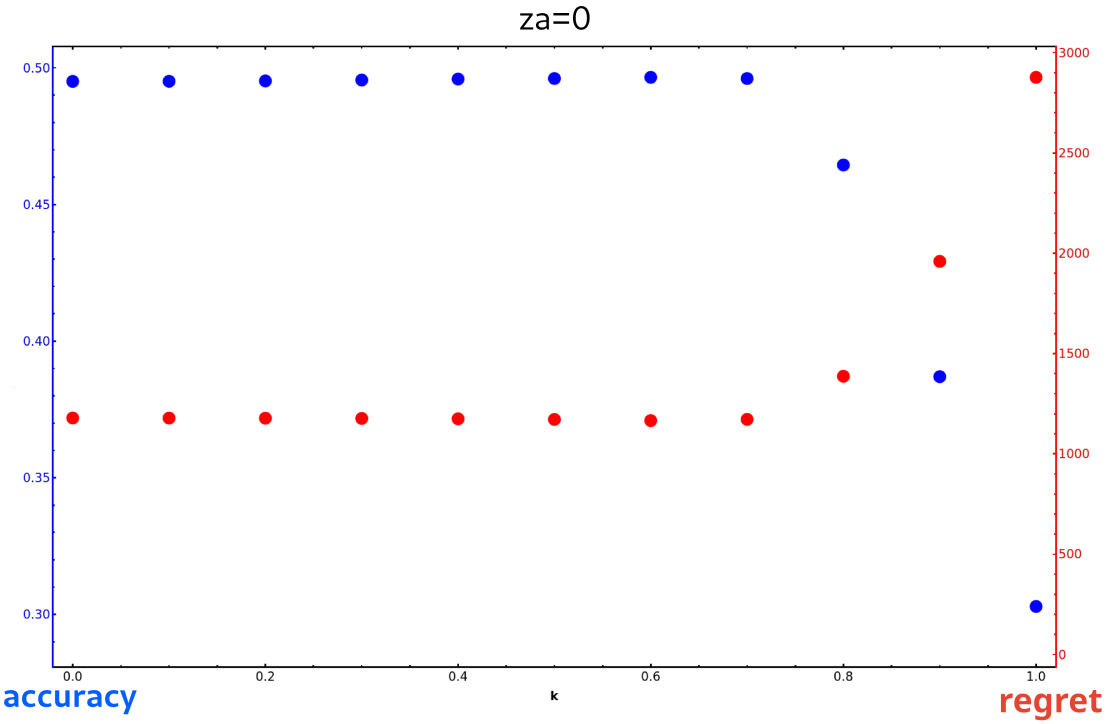


Figure 3.1: Accuracy and regret of the P_α-HDS model. Parameters: $z_A = 0$, $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $G = 4$.

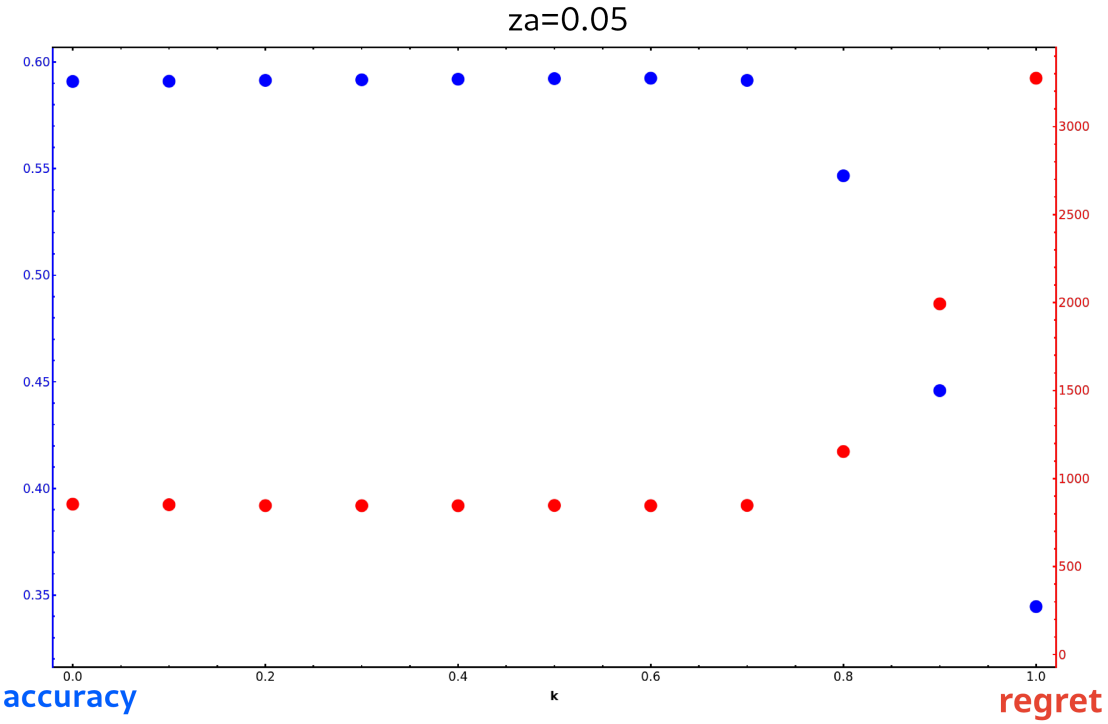


Figure 3.2: Accuracy and regret of the P_α-HDS model. Parameters: $z_A = 0.05$, $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $G = 4$.

I plot the average convergence time and the standard deviation across the values of k for $z_A = 0$ in Figure 3.3 and for $z_A = 0.05$ in Figure 3.4. From this, the convergence time and the standard deviation of the voter model is higher with respect to those of the majority, thus the latter is quicker and more accurate, as from Figures 3.1 and 3.2. Intermediate values of k show a linear trend, with the lowest value is at $k = 0$, making the majority rule model the fastest. Nonetheless, heterogeneous models show a comparable speed, especially for $k \in [0.1, 0.5]$, the convergence time and the standard deviation are approximately the same as for the majority rule.

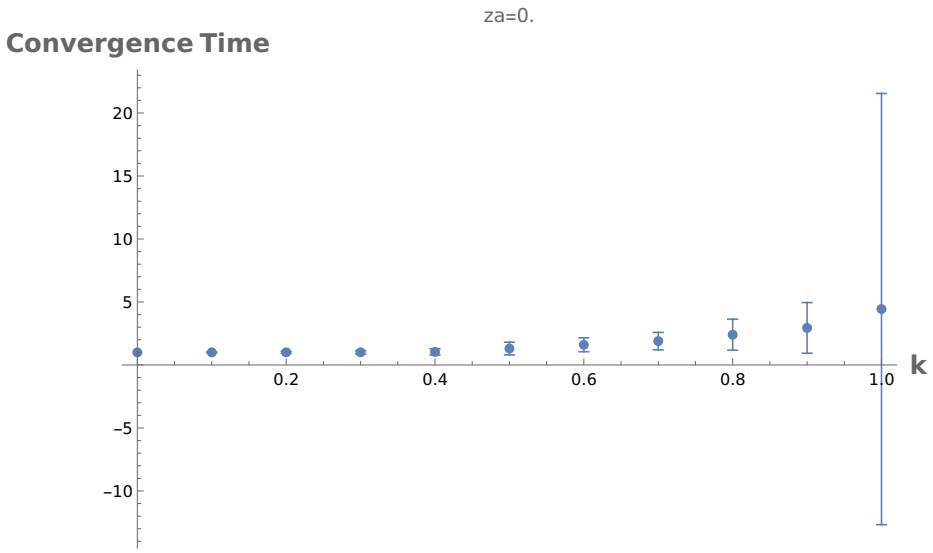


Figure 3.3: Convergence time of the P_α-HDS model. Parameters: $z_A = 0$, $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$.

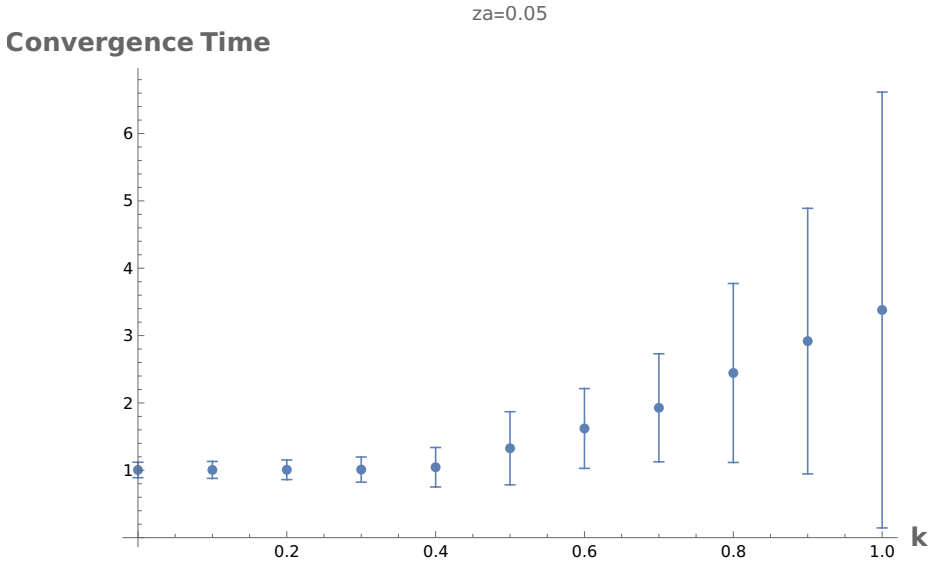


Figure 3.4: Convergence time of the P_α-HDS model. Parameters: $z_A = 0.05$, $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$.

I depict the accuracy and the cognitive cost of the P_α-HDS model, where I fix the group size to $G = 4$ and the total population to $N = 100$, while keeping $z_A = 0$ and $z_A = 0.05$ in Figure 3.5, where the points in orange represent the case for $z_A = 0$ and the points in blue represent the case for $z_A = 0.05$. The voter model has the highest cost when $z_A = 0$, but also the lowest accuracy, whereas for $z_A = 0.05$, I note a non-linearity with respect to the cognitive cost, as the heterogeneous models with $k \in [0.1, 0.9]$ have a lower cost than the pure voter model. This is due to the increasing variance of as k increases, underlined by Figure 3.4. On the other hand, the majority rule has the highest accuracy for both values of z_A and the highest cognitive cost for $z_A = 0.05$. By normalizing the cognitive cost with respect to accuracy, the optimal value of k is for $k = 0.4$ when $z_A = 0$ and when $z_A = 0.05$, as can be seen in Figure 3.6.

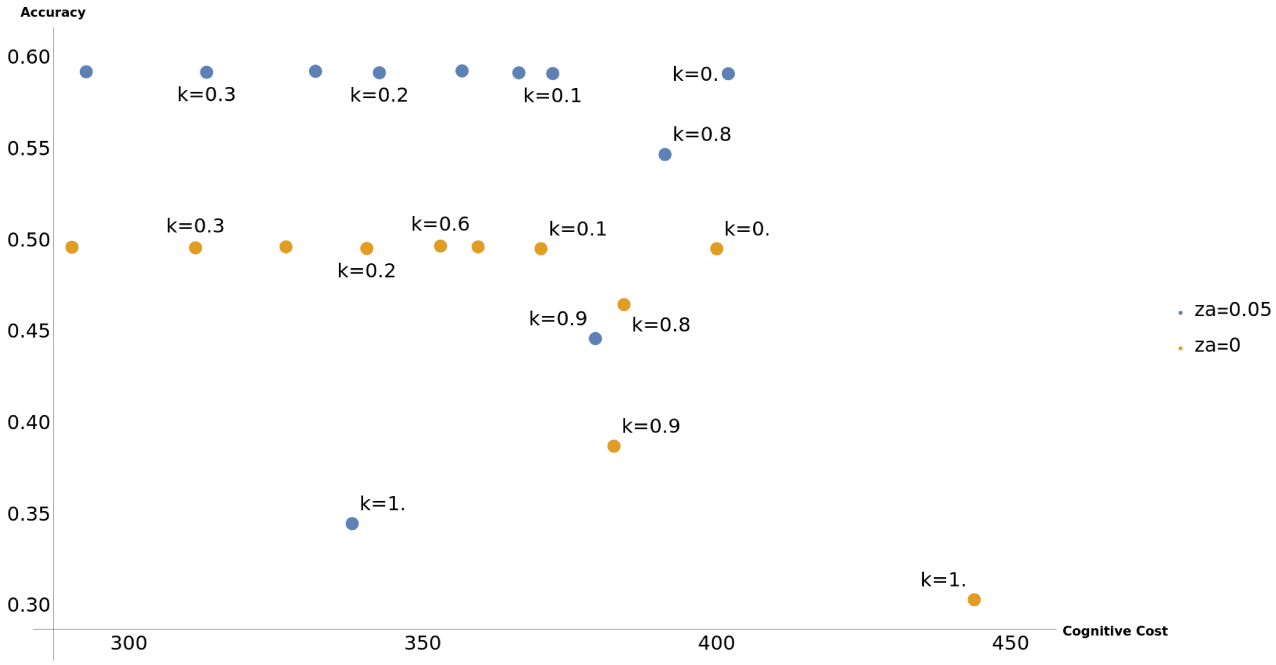


Figure 3.5: Cognitive cost versus accuracy graph for the P_α-HDS model with $z_A = 0$ in orange and $z_A = 0.05$ in blue. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $G = 4$, $N = 100$.

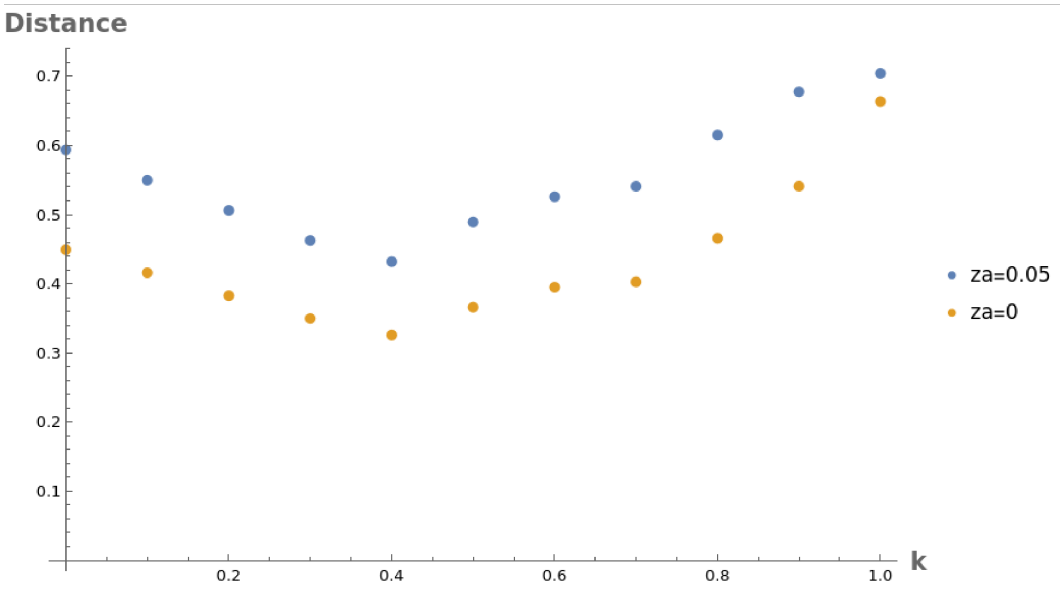


Figure 3.6: Trade-off of the P_α-HDS model between accuracy and normalized cognitive cost across the values of k. Orange points represent distance values for $z_A = 0$, while blue points represent distance values for $z_A = 0.05$. Distance represents the distance of a point in the accuracy versus normalized cognitive cost graph from the point $(0, a_{MAX})$. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $G = 4$, $N = 100$.

3.2. C-HDS

I show the accuracy and regret of the C-HDS model in function of k, when $z_A = 0$ and when $z_A = 0.05$ in Figures 3.7 and 3.8, respectively. The trend of the accuracy and of the regret of the C-HDS model resembles the trend seen in the P_α-HDS model (Figures 3.1, 3.2), with an increasing accuracy and a decreasing regret for increasing values of k. More specifically, the minimum regret is lower in the case of $z_A = 0.05$, with respect to the case of $z_A = 0$, while the maximum regret is higher.

I plot the average convergence time and its standard deviation across the values of k when $z_A = 0$ in Figure 3.9 and when $z_A = 0.05$ in Figure 3.10. From this, the convergence time and the standard deviation of the majority rule is lower with respect to those of the voter model, thus the latter is slower and less accurate, as from Figures 3.7 and 3.8. This trend is the same as that of the P_α-HDS model, as per Figures 3.3, 3.4. Intermediate values of k show a linear trend for $z_A = 0$ and a non linear trend for $z_A = 0.05$, however the lowest value is at $k = 0$, making the majority rule the fastest.

The C-HDS model does not display total dominance in the cognitive cost and accuracy plot in Figure 3.11, neither in the case of $z_A = 0$ nor in the case of $z_A = 0.05$. A point is totally dominant if and only if both its accuracy is the maximum and its cognitive cost is the minimum. Therefore, I find the optimal trade-off between accuracy and normalized cognitive cost by displaying the distance of the points in Figure 3.11 from the point that has no cognitive cost and maximum accuracy, ideally located in the top-left corner of the plot. The resulting plot is in Figure 3.12, showing that the minimum is found at $k = 0.7$ when $z_A = 0$ and when $z_A = 0.05$. As in the case of the P_α-HDS model, the same value of k that minimizes the distance, for both values of z_A. This means that the direct switch heterogeneous models do not need to adapt to different types of attacks. In particular, they do not need to adopt different percentages of voter model agents in order to have the best trade-off between cognitive cost and accuracy when different amounts of zealots with opinion A are present.

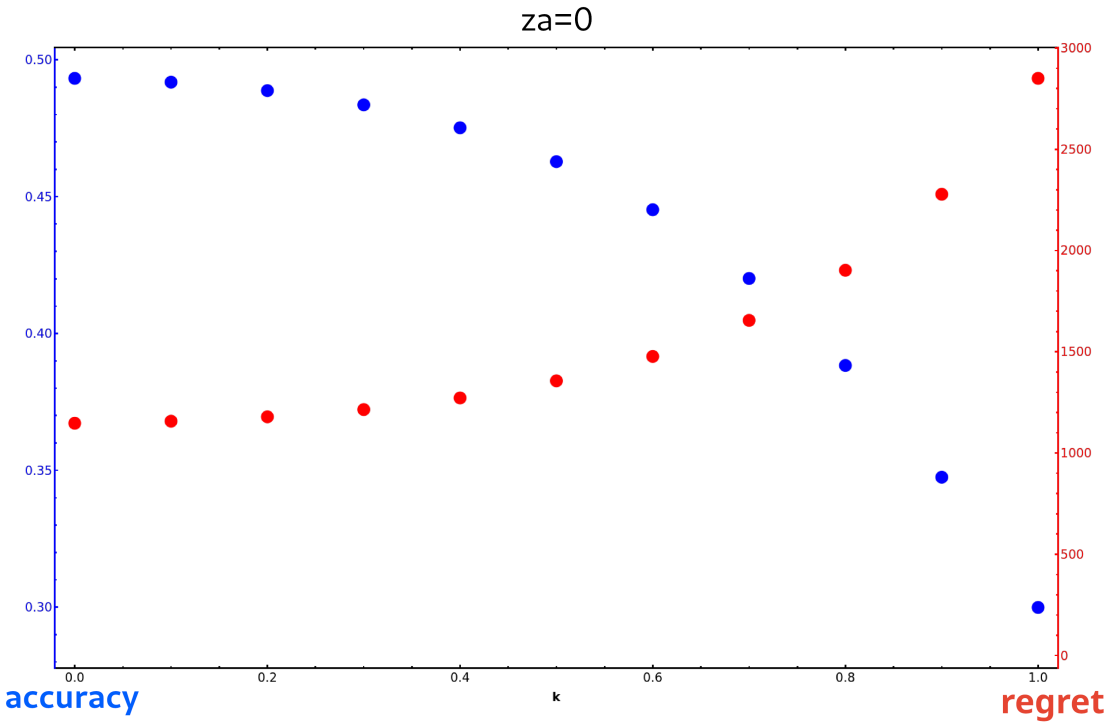


Figure 3.7: Accuracy and regret of the C-HDS model. Parameters: $z_A = 0$, $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $G = 4$.

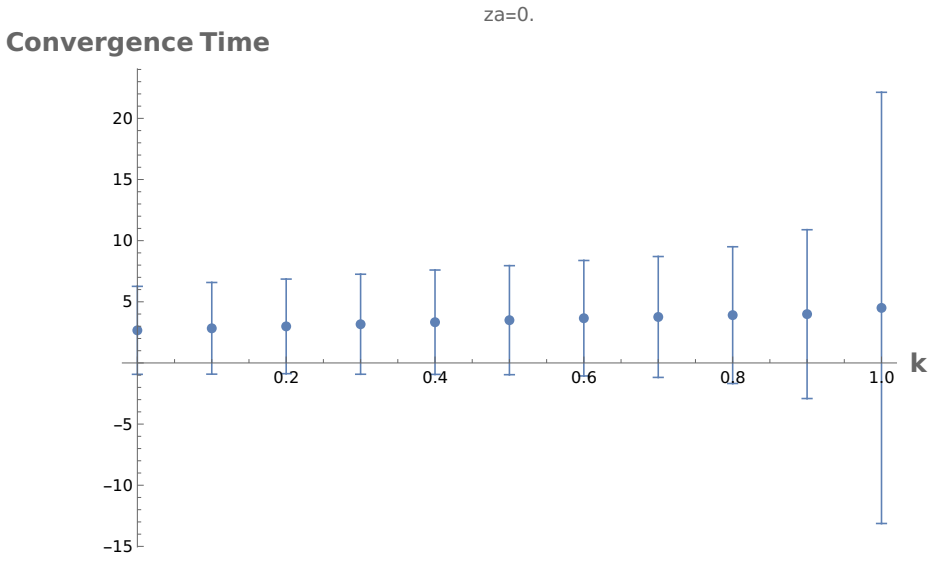


Figure 3.9: Convergence time of the C-HDS model. Parameters: $z_A = 0$, $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$.

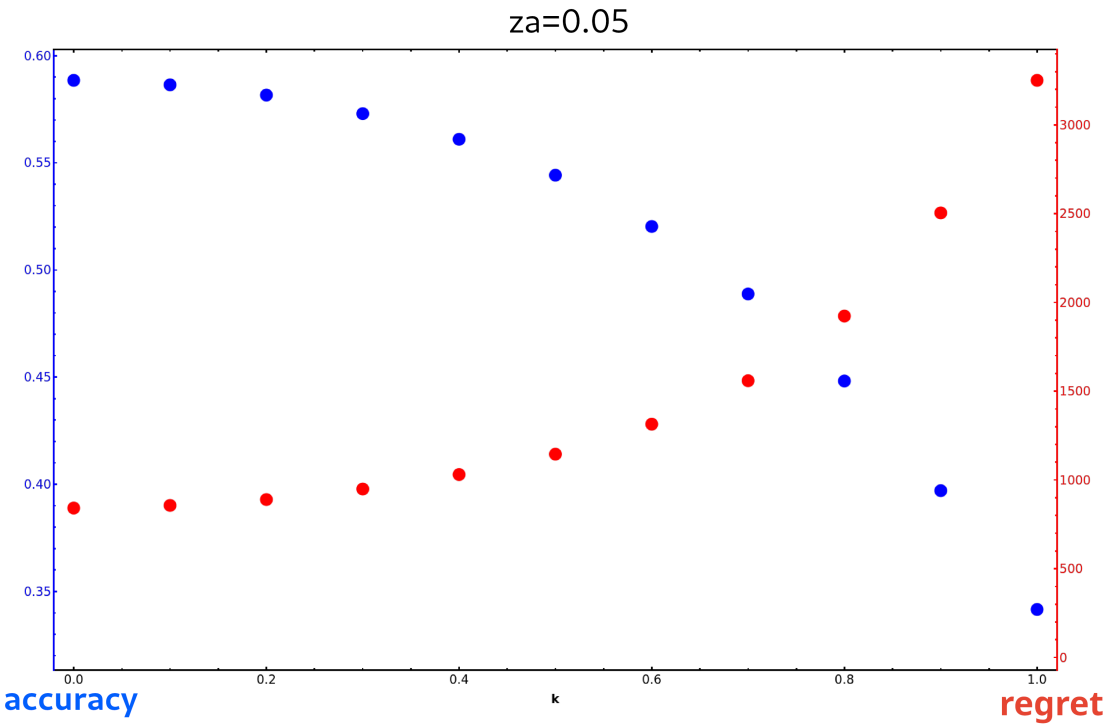


Figure 3.8: Accuracy and regret of the C-HDS model. Parameters: $z_A = 0.05$, $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $G = 4$.

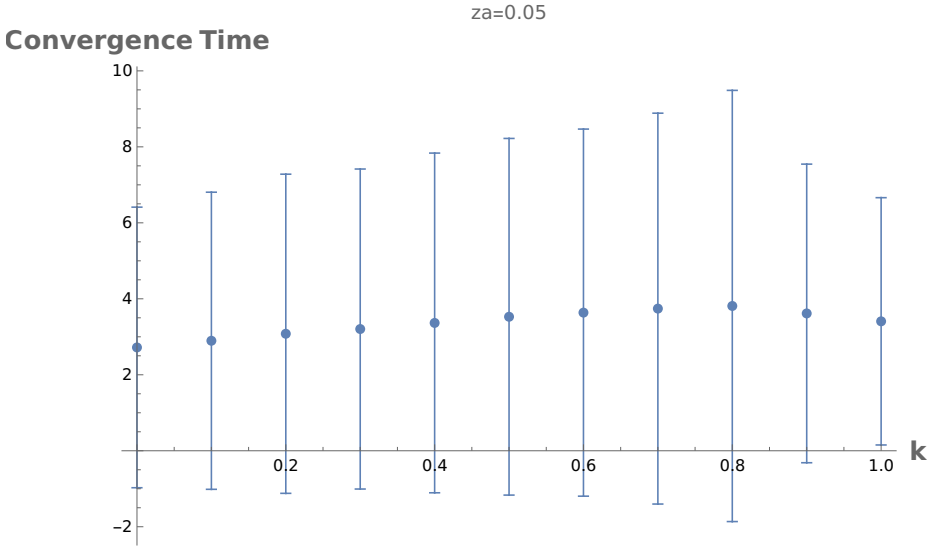


Figure 3.10: Convergence time of the C-HDS model. Parameters: $z_A = 0.05$, $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$.

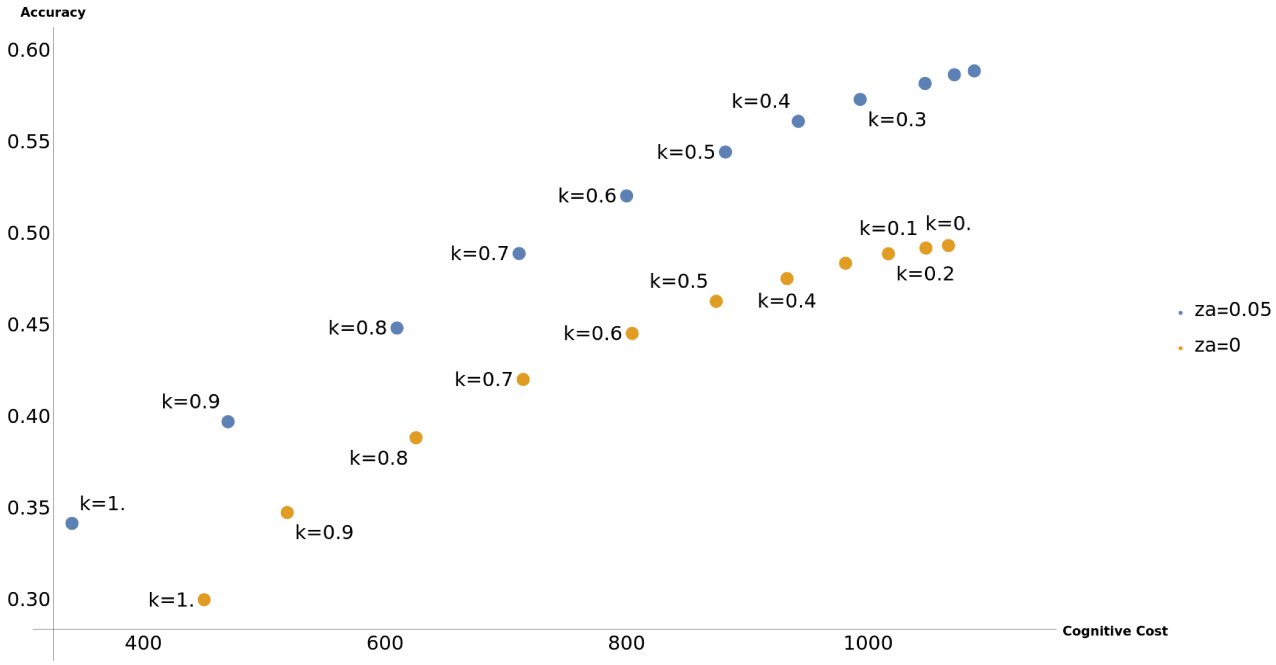


Figure 3.11: Cognitive cost versus accuracy graph for the C-HDS model with $z_A = 0$ in orange and $z_A = 0.05$ in blue. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $G = 4$, $N = 100$.

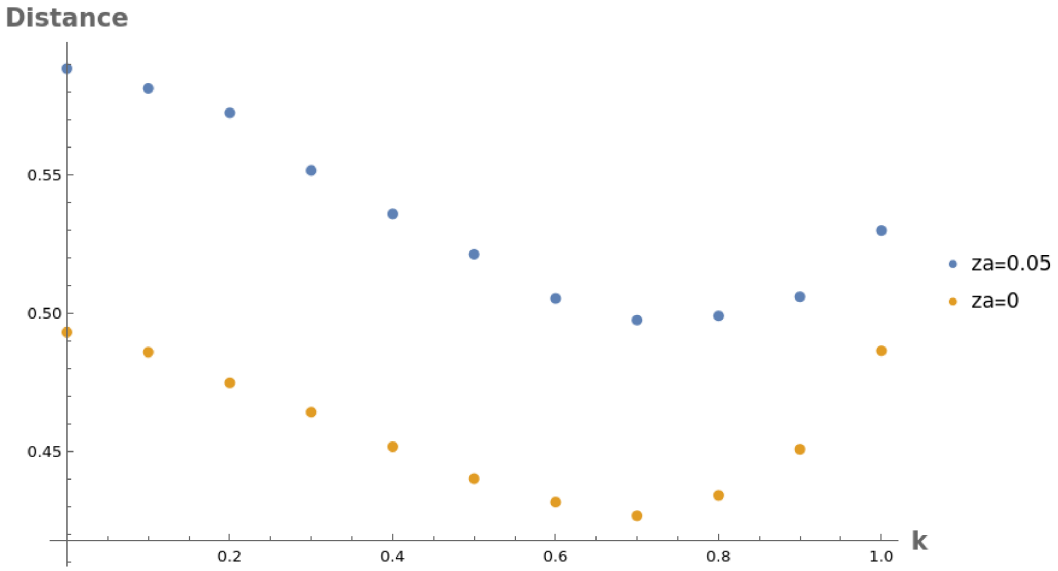


Figure 3.12: Trade-off of the C-HDS model between accuracy and normalized cognitive cost across the values of k. Orange points represent distance values for $z_A = 0$, while blue points represent distance values for $z_A = 0.05$. Distance represents the distance of a point in the accuracy versus normalized cognitive cost graph from the point $(0, a_{MAX})$. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $G = 4$, $N = 100$.

3.3. P_α-HCI

I depict the accuracy and regret of the P_α-HCI model against k, when $z_A = 0$ in Figure 3.13 and when $z_A = 0.05$ in Figure 3.14. The P_α-HCI model shows similarities with both the P_α-HDS model and the C-HDS model in terms of accuracy and regret, with a decreasing accuracy and an increasing regret as k increases. The maximum accuracy is slightly lower with respect to the other two models, however the maximum regret is significantly lower, which is the main advantage of the cross-inhibition opinion update rule.

I plot the average convergence time and its standard deviation against k when $z_A = 0$ and when $z_A = 0.05$ in Figures 3.15, 3.16, respectively. The average convergence time increases slightly, a trend present in the P_α-HDS model. Moreover, its variance follows a linear trend for both values of z_A. The variance in the case of $z_A = 0$ increases linearly as well.

I plot the cognitive cost and accuracy of the P_α-HCI model for different values of k when $z_A = 0$ (points in orange) and when $z_A = 0.05$ (points in blue) in Figure 3.17. The cognitive cost in the case of $z_A = 0$ is heavily non-linear in the range $k \in [0.3, 0.7]$, whereas it presents a non-linearity in the case of $z_A = 0.05$. Alas, there is no total domination between the points, which implies the presence of, again, a trade-off between the accuracy and the cognitive cost.

Finally, I show the distance of the normalized points in Figure 3.17 from the top left corner, with the orange points indicating the case of $z_A = 0$ and the blue points indicating the case of $z_A = 0.05$ in Figure 3.18. From this, the optimal trade-off in both the case of a wrong addressing attack and in the case of a denial of service attack is at $k = 0.2$. The fact that in both cases the same value of k attains the optimal trade-off implies that the cross-inhibition model is resilient to both attacks under the same heterogeneous configuration.

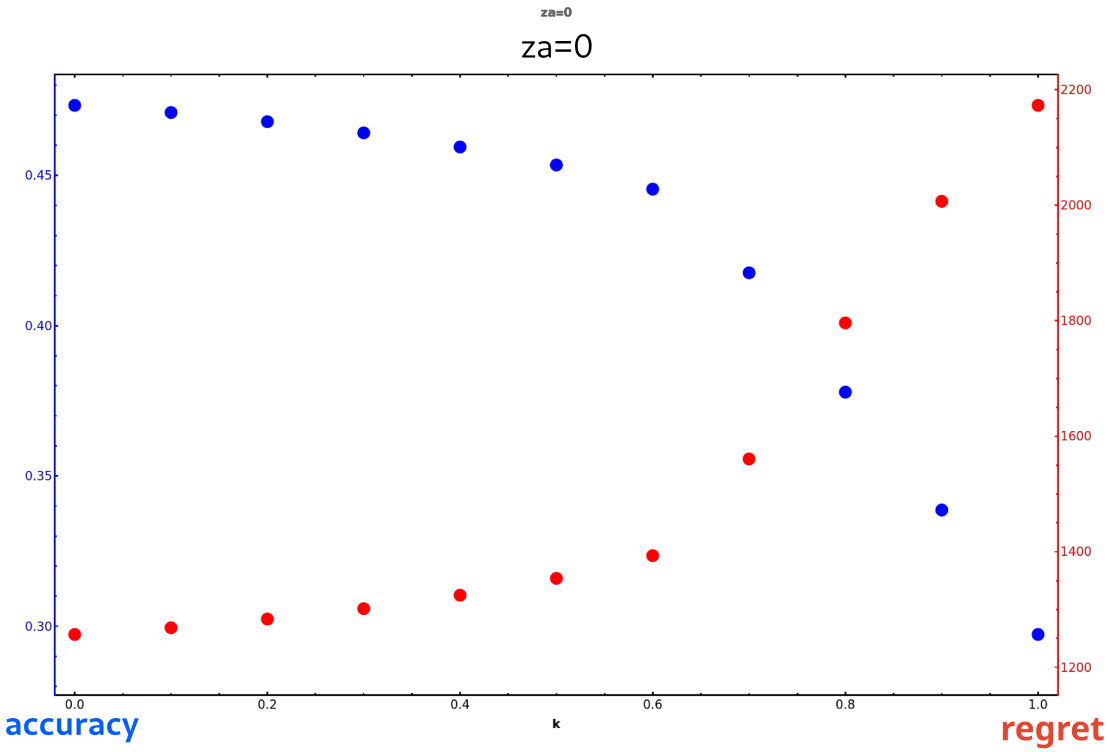


Figure 3.13: Accuracy and regret of the P_α-HCI model. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $z_A = 0$.

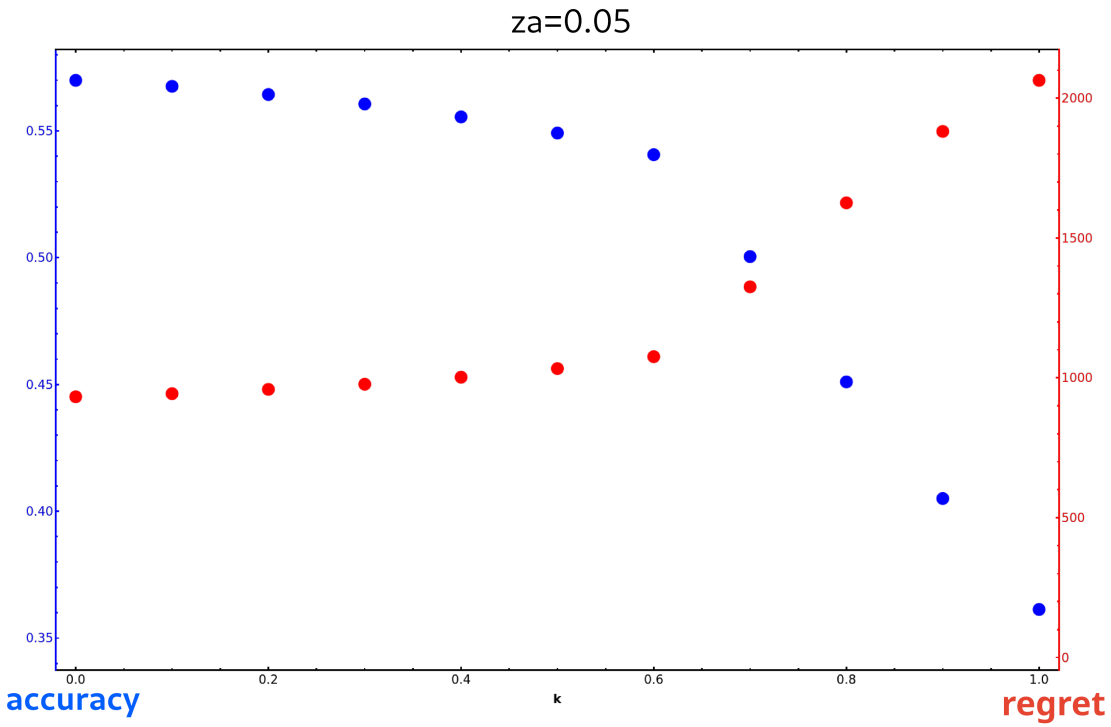


Figure 3.14: Accuracy and regret of the P_α-HCI model. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $z_A = 0.05$.

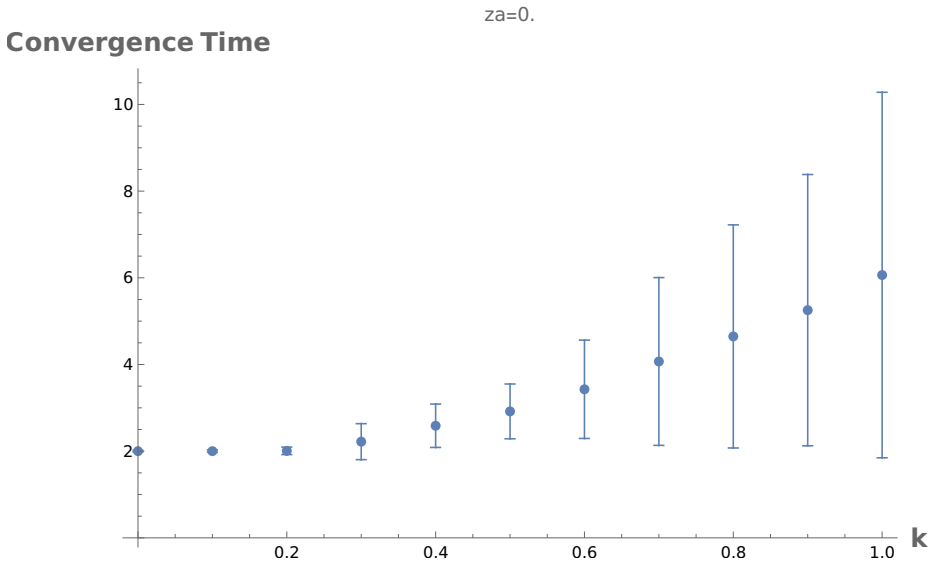


Figure 3.15: Average convergence time and standard deviation of the P_α-HCI model. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $z_A = 0$.

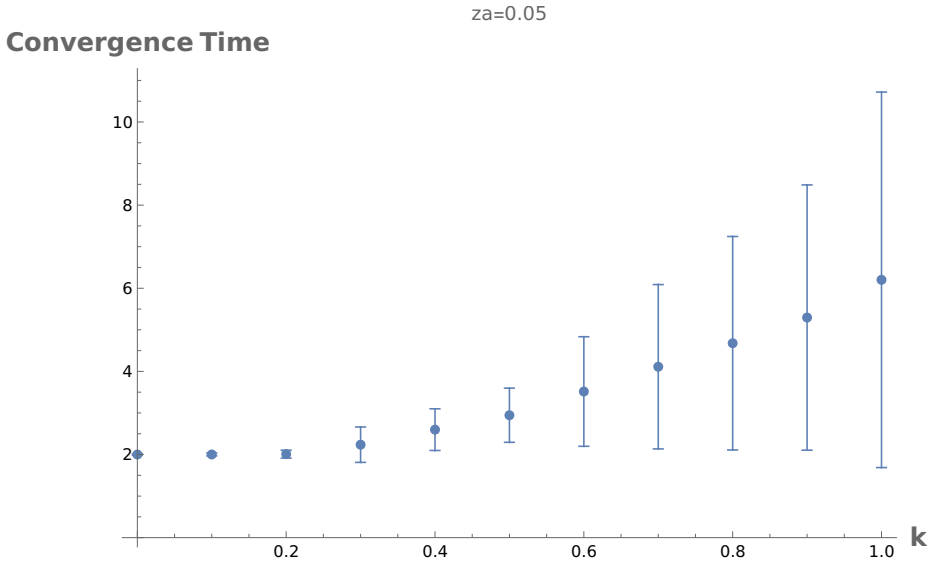


Figure 3.16: Average convergence time and standard deviation of the P_α-HCI model. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $z_A = 0.05$.

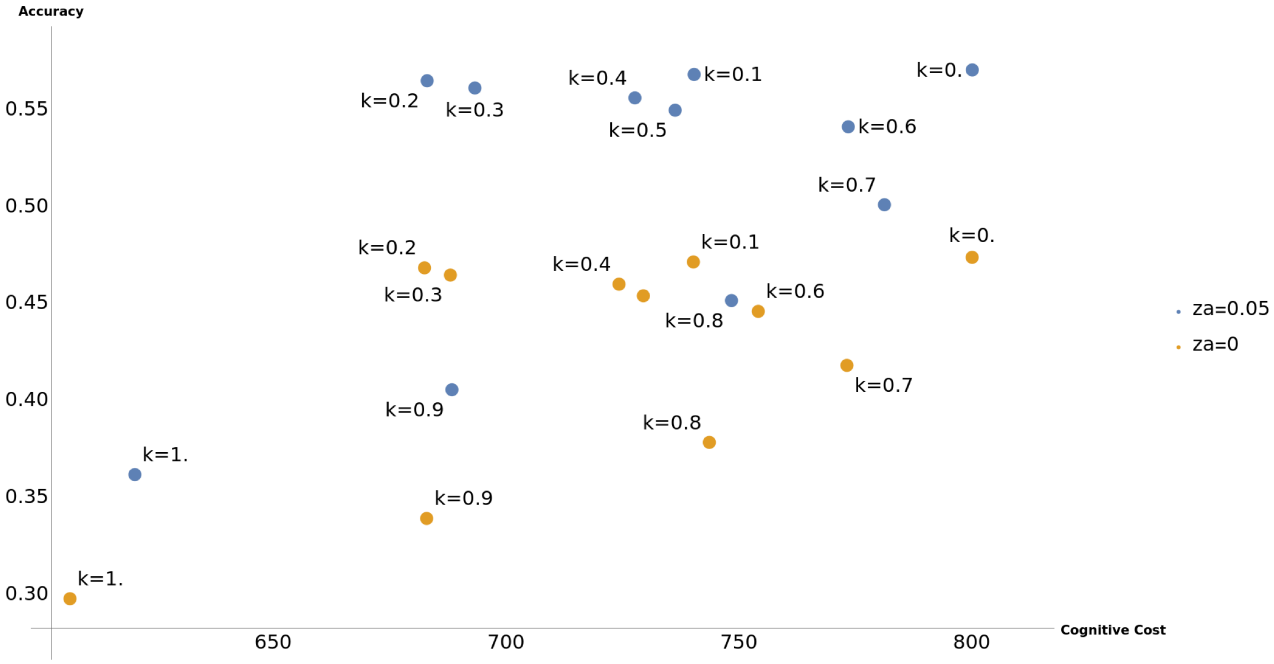


Figure 3.17: Cognitive cost and accuracy graph of the P_α-HCI model. Points in orange are for $z_A = 0$, while points in blue are for $z_A = 0.05$. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5*10^8$, $G = 4$, $N = 100$.

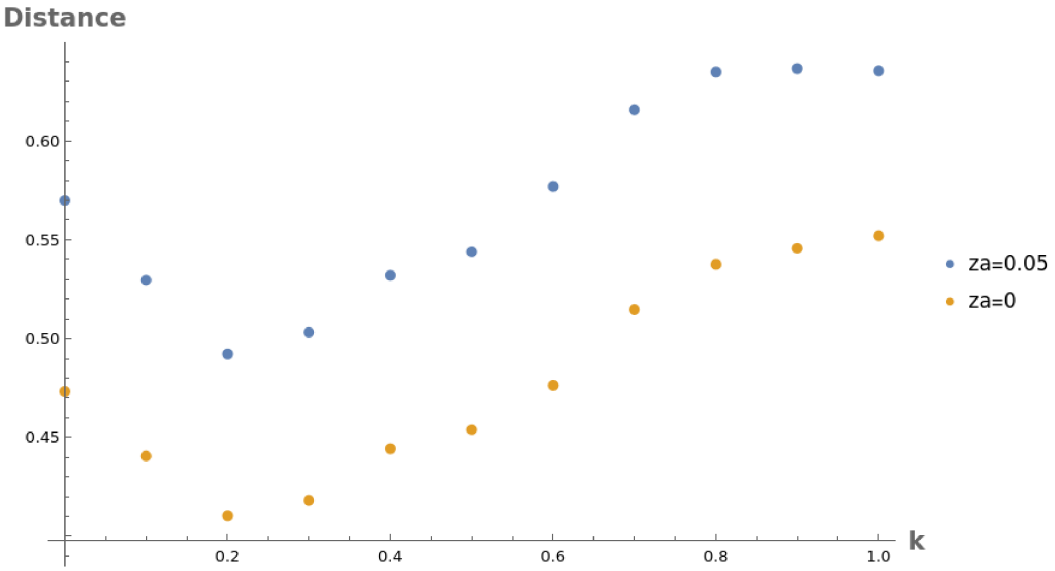


Figure 3.18: Trade-off of the P_α-HCI model between accuracy and normalized cognitive cost across the values of k. Orange points represent distance values for $z_A = 0$, while blue points represent distance values for $z_A = 0.05$. Distance represents the distance of a point in the accuracy versus normalized cognitive cost graph from the point $(0, a_{MAX})$. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $G = 4$, $N = 100$.

3.4. C-HCI

I plot the accuracy and regert of the C-HCI model against k when $z_A = 0$ and when $z_A = 0.05$ in Figures 3.19 and 3.20, respectively. The C-HCI model has the lowest maximum accuracy with respect to all the other heterogeneous models. Nevertheless, the trend is comparable to that of the P_α-HDS, C-HDS and P_α-HCI models, with a decreasing accuracy and an increasing regret as k increases.

I illustrate the average convergence time of the C-HCI model and its variance against k when $z_A = 0$ and when $z_A = 0.05$ in Figures 3.21 and 3.22, respectively. There is a linear increase in the average convergence time and in its standard deviation as k increases.

I plot the cognitive cost and the accuracy of the C-HCI model when varying k in the case of $z_A = 0$ (orange points) and in the case of $z_A = 0.05$ (blue points) in Figure 3.23. The C-HCI model does not display any total dominance neither for $z_A = 0$ nor for $z_A = 0.05$. Therefore, I plot the normalized distance of the points from the top-left corner of Figure 3.23 in function of k.

The trade-off plot in Figure 3.24 shows that the minimum distance in the normalized plot of Figure 3.23 from the top-left corner is obtained for $k = 0.8$ when $z_A = 0$ and for $k = 0.9$ when $z_A = 0.05$. The optimal trade-off is reached for different values of k, which is not the case of the P_α-HCI model (Figure 3.17).

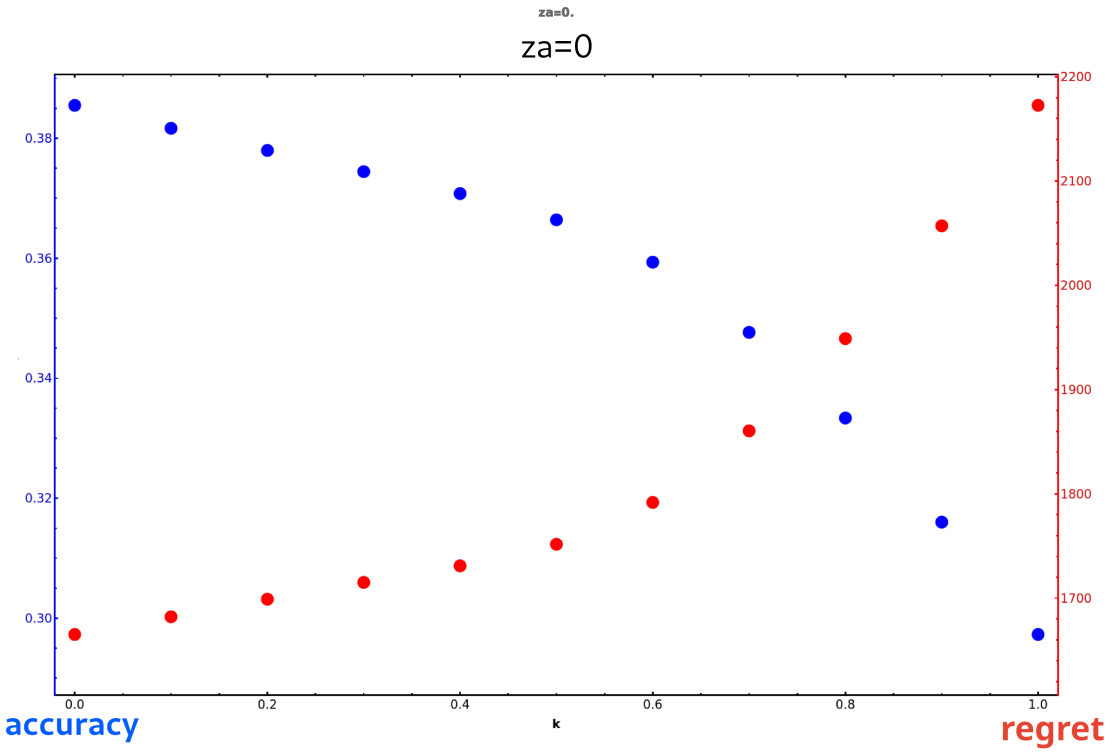


Figure 3.19: Accuracy and regret of the C-HCI model. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $z_A = 0$.

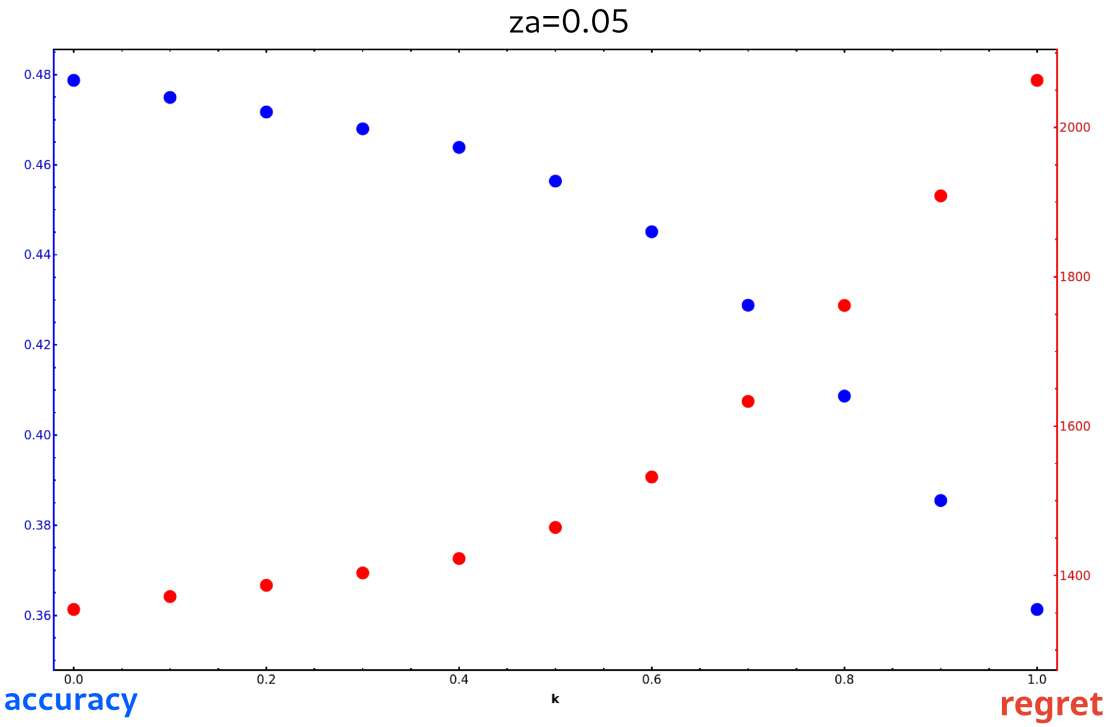


Figure 3.20: Accuracy and regret of the C-HCI model. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $z_A = 0.05$.

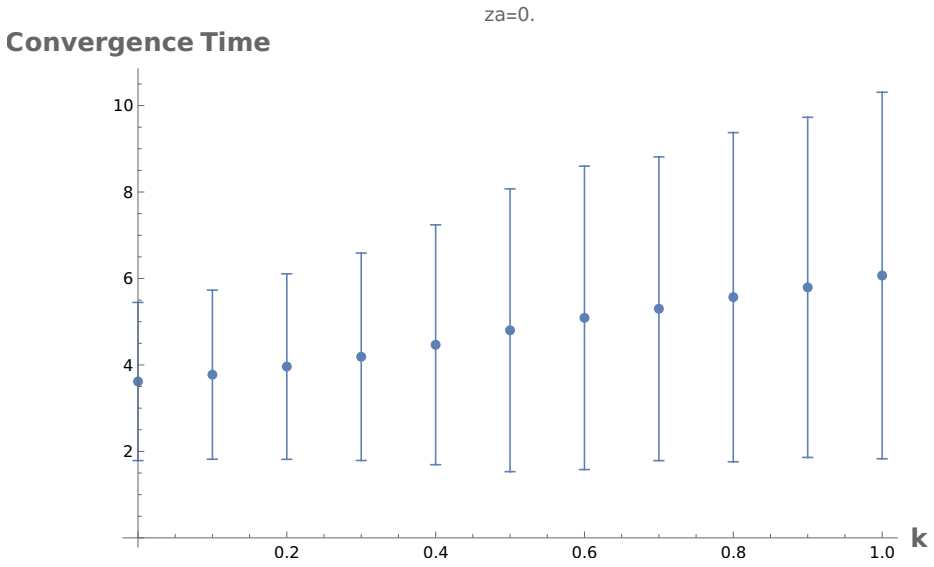


Figure 3.21: Average convergence time and standard deviation of the C-HCI model. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $z_A = 0$, $G = 4$.

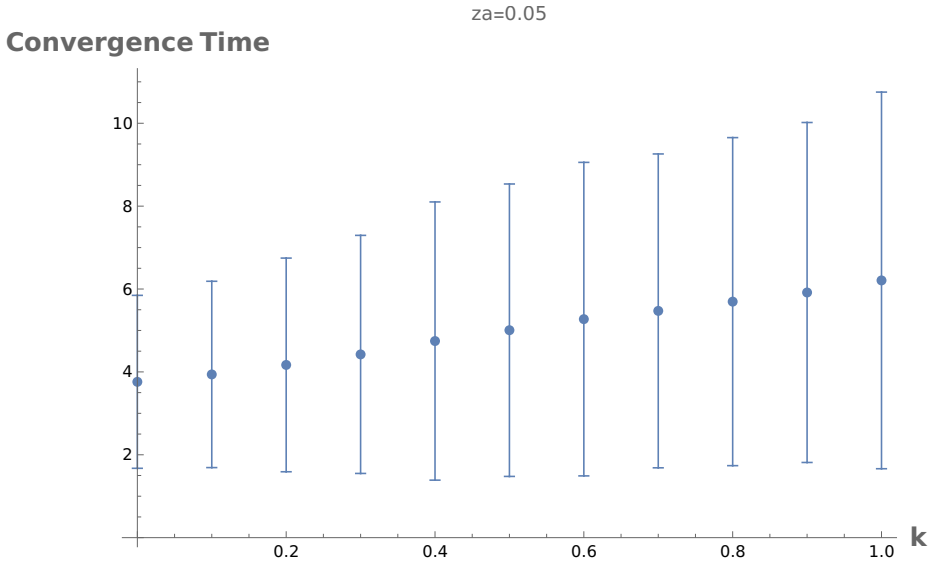


Figure 3.22: Average convergence time and standard deviation of the C-HCI model. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $z_A = 0.05$, $G = 4$.

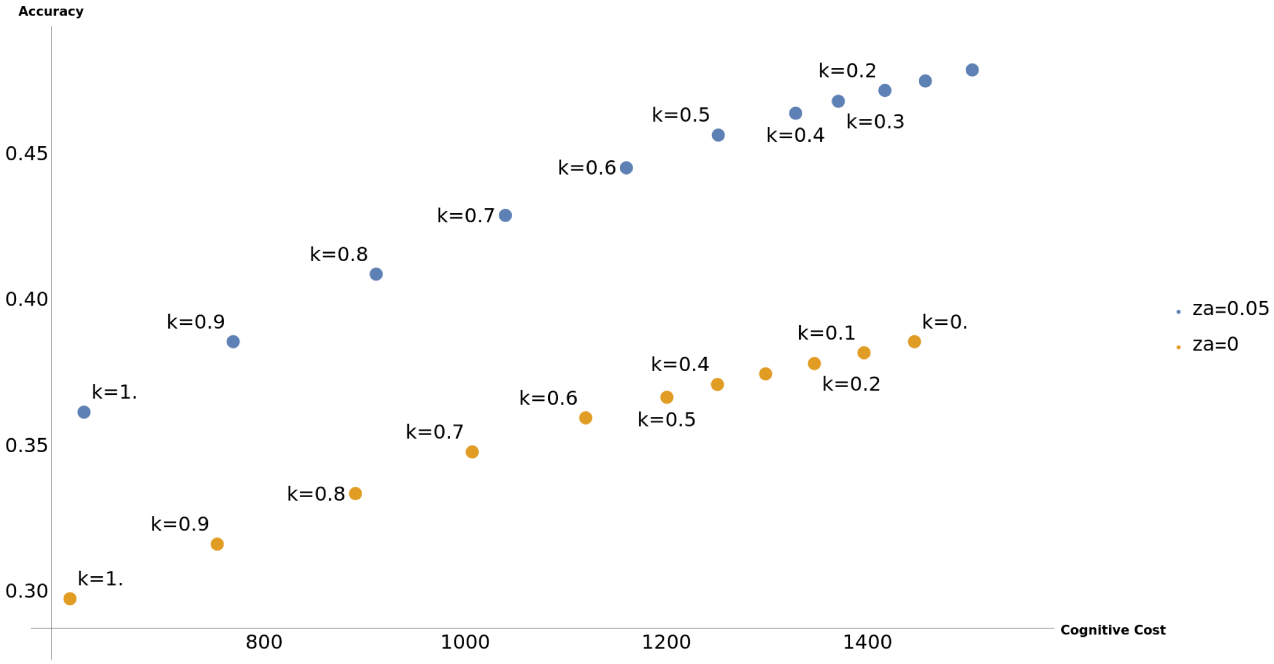


Figure 3.23: Cognitive cost and accuracy graph of the C-HCI model. Points in orange are for $z_A = 0$, while points in blue are for $z_A = 0.05$. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $G = 4$, $N = 100$.

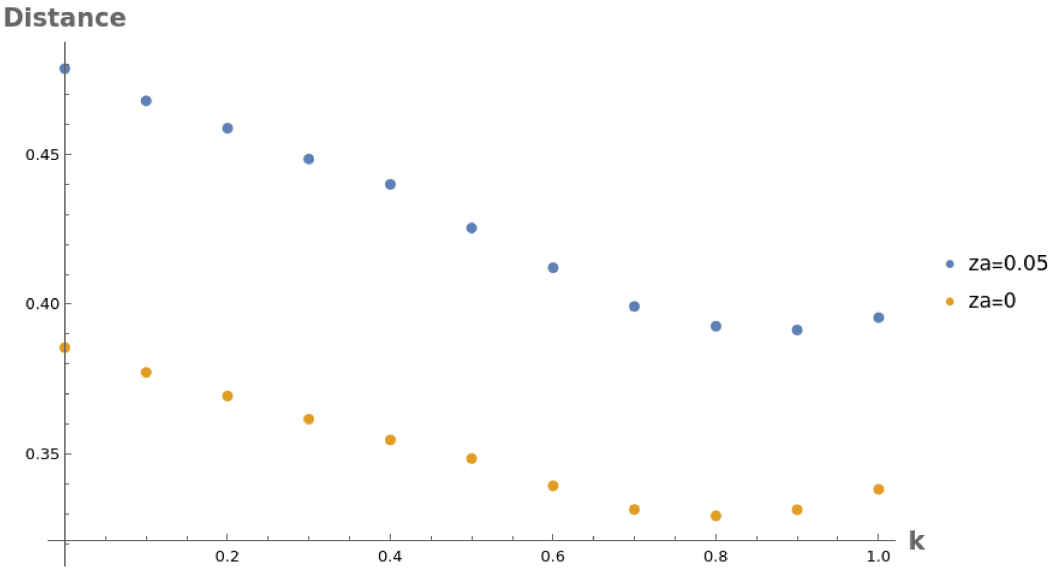


Figure 3.24: Trade-off of the C-HCI model between accuracy and normalized cognitive cost across the values of k. Orange points represent distance values for $z_A = 0$, while blue points represent distance values for $z_A = 0.05$. Distance represents the distance of a point in the accuracy versus normalized cognitive cost graph from the point $(0, a_{MAX})$. Parameters: $\epsilon = 0.7$, $t_{MAX} = 5 * 10^8$, $G = 4$, $N = 100$.

4 | Conclusion and future work

My work tackles the need for mathematical models that are able to describe adequately the complex interactions of heterogeneous decision makers within a swarm in the presence of stubborn individuals. I develop novel heterogeneous models that combine homogeneous models in order to find a trade-off between accuracy and cost. Such models are not present in the current state of the art, as shown in Section 1.3. In particular, I focus on the collective decision making process aimed at solving the best-of-n problem, by first fixing $n = 2$, and then modelling the evolution within the population brought by the exchange of opinions. The opinions have a quality associated to them, thus, by fixing one of the two options as the best one by having its quality be 1, I use the idea that an agent communicates its opinion with a frequency that is proportional to their quality [18] to model the probability of receiving a particular opinion when using a specific opinion filtering mechanism. In the case of the majority rule, I use two different probability functions for an agent to receive an opinion, while for the voter model, the two probabilities coincide. Therefore, by modelling the change within the populations, one for each opinion, I construct systems of ordinary differential equations. The numerical integration of such systems under a finite set of values of the parameters that compose them, allows to discover the behaviour of the system after convergence. Moreover, I combine the homogeneous systems while keeping the same opinion update rule, in particular I consider direct switch and cross-inhibition. Once I fix the opinion update rule, I vary the amount of voter model agents, thus creating the heterogeneous systems. In particular, since I use two different probabilities, there are two heterogeneous models for each opinion update rule. By introducing some metrics, namely accuracy, regret, average convergence time and cognitive cost, I analyze and rank the heterogeneous models in a finite set of values of the parameter k, which determines the percentage of agents using the voter model. In particular, I focus my attention on two settings: the case when no zealots with opinion A are present, which represents the wrong addressing attack where the zealots try to sway the collaborative population away from the best option, and the case when some zealots with opinion A are present ($z_A = 0.05$), which constitutes a superset of the denial of service attack, where the objective of the zealots is that of stalling the collaborative population, leading to a decision deadlock.

The analysis of the metrics leads to the conclusion that the direct switch opinion update rule, in order to have an optimal trade-off between cognitive cost and accuracy, requires the same amounts of voter model agents when different amounts of zealots A are present. In my work, these different amounts coincide with the two types of attacks, namely wrong addressing and denial of service. On the other hand, the cross-inhibition based heterogeneous models have an optimal trade-off between cognitive cost and accuracy with different values of k in the two cases.

Furthermore, the trade-off between cognitive cost and accuracy can be reached only in heterogeneous systems, as the voter model has a lower cost and a lower accuracy with respect to the majority rule, which has both a higher cost and a higher accuracy. Therefore, the possibility to fine tune the needs of a system by attaining the required accuracy while saving the cost can be guided by the mathematical modelling of heterogeneous systems.

The continuation of this work requires the use of simulations and real-life robots. I intend on continuing this research precisely in this direction, by focusing my attention first on stochastic multi-agent simulations to validate the mathematical results. The results obtained will guide me in fine tuning the models and I will be able to assess their correctness, together with any gap created by the introduction of several simplifications. Then, I will use a physics based simulator, such as ARGoS, in order to assess any differences in the behaviour of the system in a more complex environmental setup. Finally, I intend on using Kilobots, small and simple robots, to test the behaviour of the heterogeneous system in a laboratory setting, while analyzing interesting configurations of the parameters.

Bibliography

[1] T. Bose, A. Reina, and J. A. R. Marshall. Collective decision-making. *Current Opinion in Behavioral Sciences*, 16:30–34, 2017. doi: 10.1016/j.cobeha.2017.03.004.

[2] C. Castellano, S. Fortunato, and V. Loreto. Statistical physics of social dynamics. *Reviews of Modern Physics*, 81(2):591–646, 2009. doi: 10.1103/revmodphys.81.591.

[3] M. Dorigo and E. Şahin. Guest editorial. *Autonomous Robots*, 17(2):111–113, 2004. ISSN 1573-7527. doi: 10.1023/B:AURO.0000034008.48988.2b.

[4] M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, D. Burnier, A. Campo, A. L. Christensen, A. Decugniere, G. A. Di Caro, F. Ducatelle, E. Ferrante, A. Forster, J. M. Gonzales, J. Guzzi, V. Longchamp, S. Magnenat, N. Mathews, M. Montes de Oca, R. O'Grady, C. Pinciroli, G. Pini, P. Retornaz, J. Roberts, V. Sperati, T. Stirling, A. Stranieri, T. Stutzle, V. Trianni, E. Tuci, A. E. Turgut, and F. Vaussard. Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71, 2013. doi: 10.1109/MRA.2013.2252996.

[5] F. Ducatelle, G. A. Di Caro, and L. M. Gambardella. Cooperative self-organization in a heterogeneous swarm robotic system. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, page 87–94, New York, NY, USA, 2010. Association for Computing Machinery. doi: 10.1145/1830483.1830501.

[6] A. Franci, M. Golubitsky, A. Bizyaeva, and N. E. Leonard. A model-independent theory of consensus and dissensus decision making. *arXiv*, 2020.

[7] A. Franci, A. Bizyaeva, S. Park, and N. E. Leonard. Analysis and control of agreement and disagreement opinion cascades. *Swarm Intelligence*, 15(1):47–82, 2021. doi: 10.1007/s11721-021-00190-w.

[8] B. R. Johnson. Division of labor in honeybees: form, function, and proximate mechanisms. *Behavioral Ecology and Sociobiology*, 64(3):305–316, 2010. doi: 10.1007/s00265-009-0874-7.

[9] J. Krause and G. Ruxton. *Living in Groups*. Oxford Series in Ecology and Evolution. OUP Oxford, 2002.

[10] A. Ligot and M. Birattari. On using simulation to predict the performance of robot swarms. *Scientific Data*, 9(1):788, 2022. doi: 10.1038/s41597-022-01895-1.

[11] A. Reina, G. Valentini, C. Fernández-Oto, M. Dorigo, and V. Trianni. A design pattern for decentralised decision making. *PLoS ONE*, 10(10):e0140950, 2015. doi: 10.1371/journal.pone.0140950.

[12] A. Reina, T. Bose, V. Trianni, and J. A. R. Marshall. Effects of spatiality on value-sensitive decisions made by robot swarms. In R. Groß, A. Kolling, S. Berman, E. Frazzoli, A. Martinoli, F. Matsuno, and M. Gauci, editors, *Distributed Autonomous Robotic Systems: The 13th International Symposium*, pages 461–473. Springer International Publishing, Cham, 2018. doi: 10.1007/978-3-319-73008-0_32.

[13] A. Reina, R. Zakir, G. De Masi, and E. Ferrante. Cross-inhibition leads to group consensus despite the presence of strongly opinionated minorities and asocial behaviour. *arXiv*, 2211.09531, 2022. doi: 10.48550/arXiv.2211.09531.

[14] A. Reina, T.-S. Njougouo, T. Carletti, and E. Tuci. Opinion dynamics with cognitively limited individuals. *In preparation*, 2023.

[15] T. D. Seeley and S. C. Buhrman. Nest-site selection in honey bees: how well do swarms implement the "best-of-n" decision rule? *Behavioral Ecology and Sociobiology*, 49(5):416–427, 2001.

[16] V. Strobel, E. Castelló Ferrer, and M. Dorigo. Blockchain technology secures robot swarms: A comparison of consensus protocols and their resilience to byzantine robots. *Frontiers in Robotics and AI*, 7, 2020. doi: 10.3389/frobt.2020.00054.

[17] G. Valentini, E. Ferrante, H. Hamann, and M. Dorigo. Collective decision with 100 kilobots: speed versus accuracy in binary discrimination problems. *Autonomous Agents and Multi-Agent Systems*, 30(3):553–580, 2016.

[18] G. Valentini, E. Ferrante, and M. Dorigo. The best-of-n problem in robot swarms: Formalization, state of the art, and novel perspectives. *Frontiers in Robotics and AI*, 4, 2017. doi: 10.3389/frobt.2017.00009.

[19] J. C. Varughese, R. Thenius, P. Leitgeb, F. Wotawa, and T. Schmickl. A model for bio-inspired underwater swarm robotic exploration. *IFAC-PapersOnLine*, 51(2): 385–390, 2018. doi: 10.1016/j.ifacol.2018.03.066.

[20] G. Verma, A. Swami, and K. Chan. The impact of competing zealots on opinion dynamics. *Physica A: Statistical Mechanics and its Applications*, 395:310–331, 2014. doi: 10.1016/j.physa.2013.09.045.

[21] A. Waagen, G. Verma, K. Chan, A. Swami, and R. D'Souza. Effect of zealotry in high-dimensional opinion dynamics models. *Physical Review E*, 91:022811, 2015. doi: 10.1103/PhysRevE.91.022811.

[22] X. Yu, D. Saldaña, D. Shishika, and M. A. Hsieh. Resilient consensus in robot swarms with periodic motion and intermittent communication. *IEEE Transactions on Robotics*, 38(1):110–125, 2022. doi: 10.1109/TRO.2021.3088765.

A | Mathematica code

A.1. Writing data

Listing A.1: P_α-HDS

```
    relativePath = "path";
subfolder =
  "folder/sub_folder/sub_sub_folder/";

(*folder may be the name of the
model; sub_folder may be 'more_a', 'equal' or 'more_b', based on the
value of eps: in the first case, it should be +0.000001, in the
second case it should be 0 and in the third case it should be −0.000001
sub_sub_folder may represent the value of z_a, e.g. za_0 or za_0_0_5.

all these rules will be used when reading the filename*)
Do[
 Clear[Avm, Amr, Bvm, Bmr, tau];
 tol = 0.7;
 za = 0.05;
 tmax = 500000000;
 eps = −0.000001;
 awinspoints = {};
 bwinspoints = {};
 deadlockpoints = {};
 convergenceTimes = {};
 convergenceTimeSum = 0.0;
 p[x_] := 1/(1 + Exp[50 − 100*x]);
 Do[
  out =
   NDSolve[{
     Avm'[t] ==
      Bvm[t]*(Avm[t] + Amr[t] + za)/tau[t] −
       Avm[t]*(q*(Bvm[t] + Bmr[t]) + zb)/tau[t],
         Amr'[t] ==
      Bmr[t]*p[(Avm[t] + Amr[t] + za)/tau[t]] −
```

```
32          Amr[t]*p[(q*(Bvm[t] + Bmr[t]) + zb)/tau[t]],
33            tau[t] == Avm[t] + Amr[t] + q*(Bvm[t] + Bmr[t]) + za + zb,
34            Bvm[t] == k - Avm[t],
35            Bmr[t] == 1 - k - Amr[t],
36            Avm[0] == (k)/2 + eps,
37            Amr[0] == (1 - k)/2 - eps},
38        {Avm, Amr, Bvm, Bmr},
39        {t, 0, tmax}];
40     plotA = Plot[Evaluate[{Avm[t] + Amr[t]} /. out], {t, 0, tmax}];
41     pointsA = plotA // Cases[#, Line[x_] :> x, All] & // First;
42     plotB = Plot[Evaluate[{Bvm[t] + Bmr[t]} /. out], {t, 0, tmax}];
43     pointsB = plotB // Cases[#, Line[x_] :> x, All] & // First;
44     If[
45      Last[pointsA ][[2]] >= tol,
46      AppendTo[awinspoints, {zb, q}];
47      For[s = 0, s < tmax, s++, o = {Avm[s] + Amr[s]} /. out;
48       If[o[[1, 1]] >= tol, convergenceTimeSum += s;
49        AppendTo[convergenceTimes, {zb, q, s}]; Break[]]],
50      If[
51       Last[pointsB ][[2]] >= tol,
52       AppendTo[bwinspoints, {zb, q}],
53       AppendTo[deadlockpoints, {zb, q}]
54       ]];
55     ,
56     {zb, 0, 0.5, 0.005}, {q, 0, 1, 0.01}]
57   Export[
58     relativePath <> subfolder <> "convergence_time_k_" <>
59      ToString[NumberDigit[k, 0]] <> "_" <>
60      ToString[NumberDigit[k, -1]] <> "_za_" <>
61      ToString[NumberDigit[za, 0]] <> "_" <>
62      ToString[NumberDigit[za, -1]] <> "_" <>
63      ToString[NumberDigit[za, -2]] <> ".txt",
64     convergenceTimeSum/Length[awinspoints]];
65  Export[relativePath <> subfolder <> "convergence_times_k_" <>
66     ToString[NumberDigit[k, 0]] <> "_" <>
67     ToString[NumberDigit[k, -1]] <> "_za_" <>
68     ToString[NumberDigit[za, 0]] <> "_" <>
69     ToString[NumberDigit[za, -1]] <> "_" <>
70     ToString[NumberDigit[za, -2]] <> ".txt", convergenceTimes];
71  Export[relativePath <> subfolder <> "a_wins_points_k_" <>
72     ToString[NumberDigit[k, 0]] <> "_" <>
73     ToString[NumberDigit[k, -1]] <> "_za_" <>
74     ToString[NumberDigit[za, 0]] <> "_" <>
75     ToString[NumberDigit[za, -1]] <> "_" <>
```

```
    ToString[NumberDigit[za, −2]] <> ".txt", awinspoints];
  Export[relativePath <> subfolder <> "b_wins_points_k_" <>
    ToString[NumberDigit[k, 0]] <> "_" <>
    ToString[NumberDigit[k, −1]] <> "_za_" <>
    ToString[NumberDigit[za, 0]] <> "_" <>
    ToString[NumberDigit[za, −1]] <> "_" <>
    ToString[NumberDigit[za, −2]] <> ".txt", bwinspoints];
  Export[relativePath <> subfolder <> "deadlock_points_k_" <>
    ToString[NumberDigit[k, 0]] <> "_" <>
    ToString[NumberDigit[k, −1]] <> "_za_" <>
    ToString[NumberDigit[za, 0]] <> "_" <>
    ToString[NumberDigit[za, −1]] <> "_" <>
    ToString[NumberDigit[za, −2]] <> ".txt", deadlockpoints];
  Print[k]

 , {k, 0, 1, 0.1}]
```

Listing A.2: C-HDS

```
G = 4;
path = "path/folder/sub_folder/sub_sub_folder/";
(*same rules as above*)
Do[
 Clear[Avm, Amr, Bvm, Bmr, tau];
 tol = 0.7;
 za = 0;
 tmax = 500000000;
 eps = −0.000001;
 awinspoints = {};
 bwinspoints = {};
 deadlockpoints = {};
 convergenceTimes = {};
 convergenceTimeSum = 0.0;
 Do[
   out =
    NDSolve[{Amr'[t] ==
       Sum[Binomial[G, i]*(pa[t])^i*(pb[t])^(G − i), {i,
           Ceiling[G/2] + 1 − Mod[G, 2], G}]*Bmr[t] −
        Sum[Binomial[G, i]*(pa[t])^(G − i)*(pb[t])^i, {i,
           Ceiling[G/2] + 1 − Mod[G, 2], G}]*Amr[t],
      Avm'[t] == Bvm[t]*pa[t] − Avm[t]*pb[t],
          tau[t] == Amr[t] + Avm[t] + (Bmr[t] + Bvm[t])*q + za + zb,
          pa[t] == (za + Amr[t] + Avm[t])/tau[t],
          pb[t] == 1 − pa[t],
          Bvm[t] == k − Avm[t],
```

```
            Bmr[t] == 1 - k - Amr[t],
            Avm[0] == k*(0.5 + eps),
            Amr[0] == (1 - k)*(0.5 + eps)
            },
        {Avm, Amr, Bvm, Bmr},
        {t, 0, tmax}];
    plotA = Plot[Evaluate[{Avm[t] + Amr[t]} /. out], {t, 0, tmax}];
    pointsA = plotA // Cases[#, Line[x_] :> x, All] & // First;
    plotB = Plot[Evaluate[{Bvm[t] + Bmr[t]} /. out], {t, 0, tmax}];
    pointsB = plotB // Cases[#, Line[x_] :> x, All] & // First;
    If[
     Last[pointsA ][[2]] >= tol,
     AppendTo[awinspoints, {zb, q}];
     For[s = 0, s < tmax, s++, o = {Avm[s] + Amr[s]} /. out;
      If[o[[1, 1]] >= tol, convergenceTimeSum += s;
       AppendTo[convergenceTimes, {zb, q, s}]; Break[]]],
     If[
      Last[pointsB ][[2]] >= tol,
      AppendTo[bwinspoints, {zb, q}],
      AppendTo[deadlockpoints, {zb, q}]
      ]];
    ,
    {zb, 0, 0.5, 0.005}, {q, 0.01, 1, 0.01}]
   Export[
    path <> "convergence_time_k_" <> ToString[NumberDigit[k, 0]] <>
     "_" <> ToString[NumberDigit[k, -1]] <> "_za_" <>
     ToString[NumberDigit[za, 0]] <> "_" <>
     ToString[NumberDigit[za, -1]] <> "_" <>
     ToString[NumberDigit[za, -2]] <> ".txt",
    convergenceTimeSum/Length[awinspoints]];
Export[path <> "convergence_times_k_" <>
    ToString[NumberDigit[k, 0]] <> "_" <>
    ToString[NumberDigit[k, -1]] <> "_za_" <>
    ToString[NumberDigit[za, 0]] <> "_" <>
    ToString[NumberDigit[za, -1]] <> "_" <>
    ToString[NumberDigit[za, -2]] <> ".txt", convergenceTimes];
Export[path <> "a_wins_points_k_" <> ToString[NumberDigit[k, 0]] <>
    "_" <> ToString[NumberDigit[k, -1]] <> "_za_" <>
    ToString[NumberDigit[za, 0]] <> "_" <>
    ToString[NumberDigit[za, -1]] <> "_" <>
    ToString[NumberDigit[za, -2]] <> ".txt", awinspoints];
Export[path <> "b_wins_points_k_" <> ToString[NumberDigit[k, 0]] <>
    "_" <> ToString[NumberDigit[k, -1]] <> "_za_" <>
    ToString[NumberDigit[za, 0]] <> "_" <>
```

```
    ToString[NumberDigit[za, −1]] <> "_" <>
    ToString[NumberDigit[za, −2]] <> ".txt", bwinspoints];
  Export[path <> "deadlock_points_k_" <> ToString[NumberDigit[k, 0]] <>
     "_" <> ToString[NumberDigit[k, −1]] <> "_za_" <>
    ToString[NumberDigit[za, 0]] <> "_" <>
    ToString[NumberDigit[za, −1]] <> "_" <>
    ToString[NumberDigit[za, −2]] <> ".txt", deadlockpoints];
  Print[k];

 , {k, 0, 1, 0.1}]
```

Listing A.3: P_α-HCI

```
path = "path/folder/sub_folder/sub_sub_folder/";
(*same rules as above*)
Do[
 Clear[Avm, Amr, Bvm, Bmr, tau];
 tol = 0.7;
 za = 0.05;
 tmax = 500000000;
 eps = +0.000001;
 awinspoints = {};
 bwinspoints = {};
 deadlockpoints = {};
 convergenceTimes = {};
 convergenceTimeSum = 0.0;
 p[x_] := 1/(1 + Exp[50 − 100*x]);
 Do[
   out =
    NDSolve[{
      Avm'[t] ==
       Uvm[t]*(Avm[t] + Amr[t] + za)/tau[t] −
        Avm[t]*(q*(Bvm[t] + Bmr[t]) + zb)/tau[t],
          Bvm'[t] ==
       Uvm[t]*(q*(Bvm[t] + Bmr[t]) + zb)/tau[t] −
        Bvm[t]*(Avm[t] + Amr[t] + za)/tau[t],
          Amr'[t] ==
       Umr[t]*p[(Avm[t] + Amr[t] + za)/tau[t]] −
        Amr[t]*p[(q*(Bvm[t] + Bmr[t]) + zb)/tau[t]],
          Bmr'[t] ==
       Umr[t]*p[(q*(Bvm[t] + Bmr[t]) + zb)/tau[t]] −
        Bmr[t]*p[(Avm[t] + Amr[t] + za)/tau[t]],
          tau[t] == Avm[t] + Amr[t] + q*(Bvm[t] + Bmr[t]) + za + zb,
          Uvm[t] == k − Avm[t] − Bvm[t],
          Umr[t] == 1 − k − Amr[t] − Bmr[t],
```

```
            Avm[0] == k/2 + eps,
            Bvm[0] == k/2 − eps,
            Amr[0] == (1 − k)/2 + eps,
            Bmr[0] == (1 − k)/2 − eps},
        {Avm, Amr, Bvm, Bmr},
        {t, 0, tmax}];
    plotA = Plot[Evaluate[{Avm[t] + Amr[t]} /. out], {t, 0, tmax}];
    pointsA = plotA // Cases[#, Line[x_] :> x, All] & // First;
    plotB = Plot[Evaluate[{Bvm[t] + Bmr[t]} /. out], {t, 0, tmax}];
    pointsB = plotB // Cases[#, Line[x_] :> x, All] & // First;
    If[
     Last[pointsA ][[2]] >= tol,
     AppendTo[awinspoints, {zb, q}];
     For[s = 0, s < tmax, s++, o = {Avm[s] + Amr[s]} /. out;
      If[o[[1, 1]] >= tol, convergenceTimeSum += s;
       AppendTo[convergenceTimes, {zb, q, s}]; Break[]]],
     If[
      Last[pointsB ][[2]] >= tol,
      AppendTo[bwinspoints, {zb, q}],
      AppendTo[deadlockpoints, {zb, q}]
      ]];
    ,
    {zb, 0, 0.5, 0.005}, {q, 0, 1, 0.01}]
  Export[
    path <> "convergence_times_k_" <> ToString[NumberDigit[k, 0]] <>
     "_" <> ToString[NumberDigit[k, −1]] <> "_za_" <>
     ToString[NumberDigit[za, 0]] <> "_" <>
     ToString[NumberDigit[za, −1]] <> "_" <>
     ToString[NumberDigit[za, −2]] <> ".txt", convergenceTimes];
 Export[path <> "convergence_time_k_" <> ToString[NumberDigit[k, 0]] <>
     "_" <> ToString[NumberDigit[k, −1]] <> "_za_" <>
    ToString[NumberDigit[za, 0]] <> "_" <>
    ToString[NumberDigit[za, −1]] <> "_" <>
    ToString[NumberDigit[za, −2]] <> ".txt",
  convergenceTimeSum/Length[awinspoints]];
 Export[path <> "a_wins_points_k_" <> ToString[NumberDigit[k, 0]] <>
    "_" <> ToString[NumberDigit[k, −1]] <> "_za_" <>
    ToString[NumberDigit[za, 0]] <> "_" <>
    ToString[NumberDigit[za, −1]] <> "_" <>
    ToString[NumberDigit[za, −2]] <> ".txt", awinspoints];
 Export[path <> "b_wins_points_k_" <> ToString[NumberDigit[k, 0]] <>
    "_" <> ToString[NumberDigit[k, −1]] <> "_za_" <>
    ToString[NumberDigit[za, 0]] <> "_" <>
    ToString[NumberDigit[za, −1]] <> "_" <>
```

```
77    ToString[NumberDigit[za, −2]] <> ".txt", bwinspoints];
78  Export[path <> "deadlock_points_k_" <> ToString[NumberDigit[k, 0]] <>
79     "_" <> ToString[NumberDigit[k, −1]] <> "_za_" <>
80    ToString[NumberDigit[za, 0]] <> "_" <>
81    ToString[NumberDigit[za, −1]] <> "_" <>
82    ToString[NumberDigit[za, −2]] <> ".txt", deadlockpoints];
83  Print[k];
84
85  , {k, 0, 1, 0.1}]
```

Listing A.4: C-HCI

```
1  relativePath = "path";
2  subfolder = "sub_folder/sub_sub_folder/"; (*same rules as above*)
3  G = 4;
4  Do[
5   Clear[Avm, Amr, Bvm, Bmr, tau];
6   tol = 0.7;
7   za = 0.05;
8   tmax = 500000000;
9   eps = −0.000001;
10  awinspoints = {};
11  bwinspoints = {};
12  deadlockpoints = {};
13  convergenceTimeSum = 0.0;
14  convergenceTimes = List[];
15  p[x_] := 1/(1 + Exp[50 − 100∗x]);
16  Do[
17    out =
18     NDSolve[{
19       Amr'[t] ==
20        Sum[Binomial[G, i]∗(pa[t])^i∗(pb[t])^(G − i), {i,
21            Ceiling[G/2] + 1 − Mod[G, 2], G}]∗Umr[t] −
22         Sum[Binomial[G, i]∗(pa[t])^(G − i)∗(pb[t])^i, {i,
23            Ceiling[G/2] + 1 − Mod[G, 2], G}]∗Amr[t],
24       Bmr'[
25         t] == −Sum[
26            Binomial[G, i]∗(pa[t])^i∗(pb[t])^(G − i), {i,
27             Ceiling[G/2] + 1 − Mod[G, 2], G}]∗Bmr[t] +
28         Sum[Binomial[G, i]∗(pa[t])^(G − i)∗(pb[t])^i, {i,
29            Ceiling[G/2] + 1 − Mod[G, 2], G}]∗Umr[t],
30       Avm'[t] == Uvm[t]∗pa[t] − Avm[t]∗pb[t],
31           Bvm'[t] == Uvm[t]∗pb[t] − Bvm[t]∗pa[t],
32       tau[t] == Amr[t] + Avm[t] + (Bmr[t] + Bvm[t])∗q + za + zb,
33           pa[t] == (za + Amr[t] + Avm[t])/tau[t],
```

```
34            pb[t] == 1 − pa[t],
35            Uvm[t] == k − Avm[t] − Bvm[t],
36            Umr[t] == 1 − k − Amr[t] − Bmr[t],
37            Avm[0] == k/2,
38            Bvm[0] == k/2,
39            Amr[0] == (1 − k)/2 + eps,
40            Bmr[0] == (1 − k)/2 − eps},
41        {Avm, Amr, Bvm, Bmr},
42        {t, 0, tmax}];
43    plotA = Plot[Evaluate[{Avm[t] + Amr[t]} /. out], {t, 0, tmax}];
44    pointsA = plotA // Cases[#, Line[x_] :> x, All] & // First;
45    plotB = Plot[Evaluate[{Bvm[t] + Bmr[t]} /. out], {t, 0, tmax}];
46    pointsB = plotB // Cases[#, Line[x_] :> x, All] & // First;
47    If[
48     Last[pointsA ][[2]] >= tol,
49     AppendTo[awinspoints, {zb, q}];
50     For[s = 0, s < tmax, s++, o = {Avm[s] + Amr[s]} /. out;
51      If[o[[1, 1]] >= tol, convergenceTimeSum += s;
52       AppendTo[convergenceTimes, {zb, q, s}]; Break[]]],
53     If[
54      Last[pointsB ][[2]] >= tol,
55      AppendTo[bwinspoints, {zb, q}],
56      AppendTo[deadlockpoints, {zb, q}]
57      ]];
58    ,
59    {zb, 0, 0.5, 0.005}, {q, 0, 1, 0.01}]
60   Export[
61    relativePath <> subfolder <> "convergence_time_k_" <>
62     ToString[NumberDigit[k, 0]] <> "_" <>
63     ToString[NumberDigit[k, −1]] <> "_za_" <>
64     ToString[NumberDigit[za, 0]] <> "_" <>
65     ToString[NumberDigit[za, −1]] <> "_" <>
66     ToString[NumberDigit[za, −2]] <> ".txt",
67    convergenceTimeSum/Length[awinspoints]];
68  Export[relativePath <> subfolder <> "convergence_times_k_" <>
69    ToString[NumberDigit[k, 0]] <> "_" <>
70    ToString[NumberDigit[k, −1]] <> "_za_" <>
71    ToString[NumberDigit[za, 0]] <> "_" <>
72    ToString[NumberDigit[za, −1]] <> "_" <>
73    ToString[NumberDigit[za, −2]] <> ".txt", convergenceTimes];
74  Export[relativePath <> subfolder <> "a_wins_points_k_" <>
75    ToString[NumberDigit[k, 0]] <> "_" <>
76    ToString[NumberDigit[k, −1]] <> "_za_" <>
77    ToString[NumberDigit[za, 0]] <> "_" <>
```

```
    ToString[NumberDigit[za, −1]] <> "_" <>
    ToString[NumberDigit[za, −2]] <> ".txt", awinspoints];
  Export[relativePath <> subfolder <> "b_wins_points_k_" <>
    ToString[NumberDigit[k, 0]] <> "_" <>
    ToString[NumberDigit[k, −1]] <> "_za_" <>
    ToString[NumberDigit[za, 0]] <> "_" <>
    ToString[NumberDigit[za, −1]] <> "_" <>
    ToString[NumberDigit[za, −2]] <> ".txt", bwinspoints];
  Export[relativePath <> subfolder <> "deadlock_points_k_" <>
    ToString[NumberDigit[k, 0]] <> "_" <>
    ToString[NumberDigit[k, −1]] <> "_za_" <>
    ToString[NumberDigit[za, 0]] <> "_" <>
    ToString[NumberDigit[za, −1]] <> "_" <>
    ToString[NumberDigit[za, −2]] <> ".txt", deadlockpoints];
  Print[k];

  , {k, 0, 1, 0.1}]
```

A.2. Reading and visualizing data

Listing A.5: Plotting averaged accuracy, regret, convergence time and cognitive cost

```
<< "/path/plotK.wl"
name = "folder"
zName = "sub_sub_folder" (*as before*)
za = 0.05;
path1 = "path" <> name <>
    "/more_a/" <> zName <> "/";
path2 =
   "path" <> name <>
    "/equal/" <> zName <> "/";
path3 = "path" <> name <>
    "/more_b/" <> zName <> "/";
G = 4;
n = 100;
costAccuracyList = {};
m1 = ExtractK[za, path1];
m2 = ExtractK[za, path2];
m3 = ExtractK[za, path3];
accuracies = {};
regrets = {};
convergenceTimes = {};
testTimes = {};
stds = {};
```

```
costs = {};
pointsToDisplay = List[];
kStart = 0.3;
kEnd = 0.5;
 m1[[5]]
Do[
 a1 = m1[[1]][[k*10 + 1]][[2]];
 a2 = m2[[1]][[k*10 + 1]][[2]];
 a3 = m3[[1]][[k*10 + 1]][[2]];
 a = (a1 + a2 + a3)/3;
 r = (m1[[2]][[k*10 + 1]][[2]] + m2[[2]][[k*10 + 1]][[2]] +
     m3[[3]][[k*10 + 1]][[2]]) /3;
 timeList1 = m1[[5]][[k*10 + 1]];
 timeList2 = m2[[5]][[k*10 + 1]];
 timeList3 = m3[[5]][[k*10 + 1]];
 timeList = Join[timeList1, timeList2, timeList3];
 onlyTimeList = timeList[[All, 3]];
 AppendTo[accuracies, {k, a}];
 AppendTo[regrets, {k, r}];
 AppendTo[
  convergenceTimes, {k, Total[onlyTimeList]/Length[onlyTimeList]}];
 AppendTo[
  stds, {k,
   PlusMinus[Total[onlyTimeList]/Length[onlyTimeList],
    StandardDeviation[onlyTimeList]]}];
 , {k, 0, 1, 0.1}]
Graphics[
 ListPlot[accuracies, AxesLabel -> {"k", "Accuracy"},
  PlotLabel -> "za=" <> ToString[za]]]
Graphics[
 ListPlot[regrets, AxesLabel -> {"k", "Regret"},
  PlotLabel -> "za=" <> ToString[za]]]
accuracyPlot = ListPlot[
   accuracies,
   PlotStyle -> Blue,
   ImagePadding -> 100,
   Frame -> {True, True, True, False},
   FrameLabel -> {Style["k", Bold, 30], Style["Accuracy", Bold, 16]},
   FrameStyle -> {Directive[Black, Thick, FontSize -> 16],
     Directive[Blue, Thick, FontSize -> 16],
     Directive[Black, Thick, FontSize -> 16],
     Directive[Black, Thick, FontSize -> 16]},
   PlotLabel -> Style["za=" <> ToString[za], Bold, 16],
   FrameTicksStyle -> Directive[25],
```

```
    ImageSize -> Full
    ];
regretPlot = ListPlot[regrets,
    PlotStyle -> Red,
    ImagePadding -> 100,
    Axes -> False,
    Frame -> {False, False, False, True},
    FrameLabel -> {{"h", Style["Regret", Bold, 16]}, {"h", "H"}},
    PlotLabel -> Style["za=" <> ToString[za], Bold, 16],
    FrameStyle -> {Directive[Black, Thick, FontSize -> 16],
       Directive[Black, Thick, FontSize -> 16],
       Directive[Black, Thick, FontSize -> 16],
       Directive[Red, Thick, FontSize -> 16]},
    FrameTicks -> {{None, All}, {None, None}},
    ImageSize -> Full,
    TicksStyle -> Large,
    FrameTicksStyle -> Directive[25]];
ImageCompose[accuracyPlot, regretPlot]
Graphics[
 ListPlot[stds,
  AxesLabel -> {Style["k", Bold, 16],
    Style["Convergence Time", Bold, 16]},
  PlotLabel -> "za=" <> ToString[za], PlotRange -> Full,
  ImageSize -> Large]]
speedAccuracyList = {};
Do[
  AppendTo[speedAccuracyList,
   Labeled[ {convergenceTimes[[i ]][[2]], accuracies [[ i ]][[2]]},
    Text[Style["k=" <> ToString[i*0.1 - 0.1], Bold, 16]]]]
  , {i, 1, Length[accuracies]}];
Graphics[
  ListPlot[speedAccuracyList,
   AxesLabel -> {Style["Convergence Time", Bold, 16],
     Style["Accuracy", Bold, 16]},
   PlotLabel -> "za=" <> ToString[za], ImageSize -> Full]];
  Do[
    cost =
   n*(stds[[k*10 + 1]][[2]][[1]] (*+stds[[k *10+1]][[2]][[
     2]]*))*(k + (1 - k)*G);
  AppendTo[costs, cost];
  AppendTo[costAccuracyList,
   Labeled[{cost, accuracies [[Floor[10*k] + 1]][[2]]},
    Text["k=" <> ToString[k]]]], {k, 0, 1, 0.1}];
Graphics[
```

```
111 ListPlot[costAccuracyList,
112  AxesLabel -> {Style["Cognitive␣Cost", Bold, 16],
113    Style["Accuracy", Bold, 16]}, ImageSize -> Full,
114  PlotRange -> Full, LabelStyle -> Directive[Black, Large]]]
115 MinDistance[costAccuracyList, {0, 0.5}]
116 Export[path <> name <>
117  "/costacc_" <> zName <> ".txt", costAccuracyList]
118 N[costAccuracyList]
119 maxAccuracy =
120  N[Max[Take[Flatten[accuracies], {2, Length[accuracies]*2, 2}]]];
121 maxCost = Max[costs];
122 alfa = maxAccuracy/maxCost;
123 minDist = +Infinity;
124 minK = 1;
125 distances = {}
126 Do[
127 d = Sqrt[(alfa*costs[[i]])^2 + (maxAccuracy - accuracies[[i, 2]])];
128 AppendTo[distances, {N[(i - 1)/10], d}];
129 If[d < minDist, minDist = d; minK = N[(i - 1)/10]]
130 , {i, 1, Length[costs]}]
131 minDist
132 minK
133 distances
134 Export[path <> name <>
135  "/distances_" <> zName <> ".txt", distances]
136 ListPlot[distances,
137 AxesLabel -> {Style["k", Bold, 16], Style["Distance", Bold, 16]},
138 ImageSize -> Large,
139 PlotRange -> Full]
```

Listing A.6: Plotting the condensed cost/accuracy and distance graphs for $z_A = 0, 0.05$

```
1  icaZa005 = Import["path" <> name <> "/costacc_za_0_0_5.txt"];
2  caZa005 = ToExpression[StringSplit[ica, "\n"]];
3  dZa005 =
4    ToExpression[
5     StringSplit[ Import["path" <> name <> "/distances_za_0_0_5.txt"],
6      "\n"]];
7  icaZa0 = Import["path" <> name <> "/costacc_za_0.txt"];
8  caZa0 = ToExpression[StringSplit[icaZa0, "\n"]];
9  dZa0 = ToExpression[
10    StringSplit[ Import["path" <> name <> "/distances_za_0.txt"],
11     "\n"]];
12 p = Rasterize[
13   ListPlot[{caZa005, caZa0},
```

```
    AxesLabel -> {Style["Cognitive Cost", Bold, 16],
      Style["Accuracy", Bold, 16]}, ImageSize -> Full,
    PlotRange -> Full, LabelStyle -> Directive[Black, Large] ,
    PlotLegends -> {"za=0.05", "za=0"}]]
p1 = Rasterize[ListPlot[{dZa005, dZa0},
    AxesLabel -> {Style["k", Bold, 16], Style["Distance", Bold, 16]},
    ImageSize -> Large,
    PlotRange -> Full,
     PlotLegends -> {"za=0.05", "za=0"}]]
```

Listing A.7: Plotting the parameter space of a heterogeneous model, showing the change for 3 values of k

```
path = "";
za = 0.0;
awinspoints = {};
bwinspoints = {};
deadlockpoints = {};
pointsToDisplay = List [];
Do[
  awinspoints =
   ToExpression[
    StringSplit [
     Import[path <> "a_wins_points_k_" <>
       ToString[NumberDigit[k, 0]] <> "_" <>
       ToString[NumberDigit[k, -1]] <> "_za_" <>
       ToString[NumberDigit[za, 0]] <> "_" <>
       ToString[NumberDigit[za, -1]] <> "_" <>
       ToString[NumberDigit[za, -2]] <> ".txt"], "\n"]] ;
  bwinspoints =
   ToExpression[
    StringSplit [
     Import[path <> "b_wins_points_k_" <>
       ToString[NumberDigit[k, 0]] <> "_" <>
       ToString[NumberDigit[k, -1]] <> "_za_" <>
       ToString[NumberDigit[za, 0]] <> "_" <>
       ToString[NumberDigit[za, -1]] <> "_" <>
       ToString[NumberDigit[za, -2]] <> ".txt"], "\n"]];
  deadlockpoints =
   ToExpression[
    StringSplit [
     Import[path <> "deadlock_points_k_" <>
       ToString[NumberDigit[k, 0]] <> "_" <>
       ToString[NumberDigit[k, -1]] <> "_za_" <>
       ToString[NumberDigit[za, 0]] <> "_" <>
```

```
33          ToString[NumberDigit[za, -1]] <> "_" <>
34          ToString[NumberDigit[za, -2]] <> ".txt"], "\n"]];
35    AppendTo[
36     pointsToDisplay, {awinspoints, bwinspoints, deadlockpoints, k}];
37    , {k, 0.8, 1, 0.1}];
38 awinsPointsK08 = pointsToDisplay[[1, 1]];
39 bwinsPointsK08 = pointsToDisplay[[1, 2]];
40 deadlockPointsK08 = pointsToDisplay[[1, 3]];
41 awinsPointsK09 = pointsToDisplay[[2, 1]];
42 bwinsPointsK09 = pointsToDisplay[[2, 2]];
43 deadlockPointsK09 = pointsToDisplay[[2, 3]];
44 awinsPointsK1 = pointsToDisplay[[3, 1]];
45 bwinsPointsK1 = pointsToDisplay[[3, 2]];
46 deadlockPointsK1 = pointsToDisplay[[3, 3]];
47 commonAPoints = {};
48 commonBPoints = {};
49 commonDeadlockPoints = {};
50 firstDeadlockPointsBottom = {}; (*points that are associated to the victory of A in k=0.9 but
       resulted in a deadlock in k=1*)
51 secondDeadlockPointsBottom = {};(*points that are associated to the victory of A in k=0.8 but
       resulted in a deadlock in k=0.9*)
52 firstDeadlockPointsTop = {};(*points that are associated to the victory of B in k=0.9 but resulted in
       a deadlock in k=1*)
53 secondDeadlockPointsTop = {};(*points that are associated to the victory of B in k=0.8 but resulted
       in a deadlock in k=0.9*)
54 Do[
55  If[MemberQ[ awinsPointsK08, {zb, q}] &&
56    MemberQ[awinsPointsK1, {zb, q}], AppendTo[commonAPoints, {zb, q}],
57   If[MemberQ[ bwinsPointsK08, {zb, q}] &&
58     MemberQ[bwinsPointsK09, {zb, q}] &&
59     MemberQ[ bwinsPointsK1, {zb, q}],
60    AppendTo[commonBPoints, {zb, q}],
61    If[MemberQ[ deadlockPointsK08, {zb, q}] &&
62      MemberQ[deadlockPointsK09, {zb, q}] &&
63      MemberQ[ deadlockPointsK1, {zb, q}],
64     AppendTo[commonDeadlockPoints, {zb, q}],
65     If[MemberQ[ awinsPointsK08, {zb, q}] &&
66       MemberQ[deadlockPointsK09, {zb, q}],
67      AppendTo[secondDeadlockPointsBottom, {zb, q}],
68      If[MemberQ[ awinsPointsK09, {zb, q}] &&
69        MemberQ[ deadlockPointsK1, {zb, q}],
70       AppendTo[firstDeadlockPointsBottom, {zb, q}],
71       If[
72        MemberQ[ bwinsPointsK08, {zb, q}] &&
```

```
          MemberQ[ deadlockPointsK09, {zb, q}],
         AppendTo[secondDeadlockPointsTop, {zb, q}],
         AppendTo[firstDeadlockPointsTop, {zb, q}]
         ]

       ]
       ]]]]
 , {zb, 0, 0.5, 0.005}, {q, 0.01, 1, 0.01}]
ListPlot[{commonAPoints, commonBPoints, commonDeadlockPoints,
  secondDeadlockPointsBottom, firstDeadlockPointsBottom,
  secondDeadlockPointsTop, firstDeadlockPointsTop},
 PlotStyle -> {Blue, Red, Yellow, Green, RGBColor[1, 0.5, 1],
   RGBColor[0, 0.5, 0], RGBColor[1, 0, 1]},
 PlotLabel -> "za=" <> ToString[za], AxesLabel -> {"zb", "q"},
 ImageSize -> Large]
```

List of Figures

Acknowledgements

I would like to thank prof. Francesco Amigoni, prof. Marco Dorigo, Andreagiovanni Reina and Raina Zakir for allowing me to work on this topic and giving me the opportunity to work at the IRIDIA laboratory. It has been a delightful experience and it allowed me to find my path.

I want to thank my mom for allowing me to live my best life, supporting me constantly and making me the person I am today. Without you, I would be nothing and I love you with all my heart.

I feel a deep need to thank my grandparents, who have raised me to their best capabilities and who have taught me many lessons about life.

I would like to thank the rest of my family for the unconditioned support and love they have given me, I love you all and I hope I will be able to spend more time with you.

The list of people I consider friends, or I have considered as such, is extremely long. This is not an easy task for me to tackle, but know that if you are reading this and I haven't mentioned you, you are very special to me in some way. Thank you for being there for me: Alessandro B, Alessio DN, Alessio P, Angelo M, Antonio R, Chiara A, Emanuele B, Francesca E, Francesca T, Francesco C, Francesco F, Giorgio DT, Ilaria DE, Letizia R, Leonardo, Leonardo S, Lorenzo A, Lorenzo DS, Luca M, Martina B, Matteo M, Nicolas C, Paola DA, René B, Riccardo B, Samuele C, Stefano L, Valeria C.