POLITECNICO DI MILANO

Facoltà di Ingegneria dell'Informazione

Corso di Laurea Specialistica in
Ingegneria Informatica

Group Size Estimation in Swarm Robotics

Relatore: Prof. Marco COLOMBETTI

Correlatori: Prof. Marco DORIGO
Dott. Mauro BIRATTARI
Ing. Carlo PINCIROLI

Tesi di Laurea di:

Manuele BRAMBILLA
Matr. 708295

Anno Accademico 2008 - 2009

Abstract

This study proposes three methods that let individual robots in a group estimate the size of the group in a distributed manner. The process is loosely based on the signaling behavior of fireflies and crickets. The first method is a modification of the work of Melhuish et al. based on local robot signaling to estimate the group's size. In Melhuish et al. method, each robot emits a signal and can perceive the signals of neighboring robots in close proximity. By counting the received signals, each robot can derive an estimate of the size of the group. We modify this method so as to sensibly stabilize the output. The second method follow the dual approach, in which each robot consider the time elapsed between one received signal and another. Though the estimate obtained by this method is noisier, the results are obtained in a very short time making this technique more suitable for larger groups. Eventually we propose a third estimation technique based on more explicit communication. Each robot communicates to its neighbors the perceived size of the group. This method proves to be more or as accurate as the first one in a much shorter time. All methods are tested and analysed with estensive simulated experiments.

Abstract

La swarm robotics è un ramo della robotica collettiva incentrata sullo studio di grandi gruppi di robot. Tali sistemi presentano un alto livello di parallelismo e ridondanza rendendoli quindi ideali per scenari complessi e ad alto livello di rischio come missioni di soccorso o l'esplorazione spaziale. In tali applicazioni, è naturale aspettarsi malfunzionamenti di singoli robot. Tenere traccia del numero di robot attivi è un'informazione importante per regolare le azioni di uno swarm. Inoltre, esistono casi in cui un numero specifico di robot garantisce performace ottimali.

Nel mio lavoro di tesi, ho presentato tre algoritmi distribuiti che permettono a ciuascun robot nello swarm di stimare, individualmente e in maniera affidabile, la dimensione del gruppo. I primi si basano sull'assunzione minimale che ogni robot possieda un sistema di comunicazione base: ogni robot puo' inviare segnali ai propri vicini entro un raggio molto limitato. L'ultimo metodo rimuove questo limite e considera robot in grado di inviare un semplice messaggio in modo da ottenere risultati migliori e in minor tempo.

Il primo algoritmo è ispirato dal lavoro di group size estimation, unico esempio in letteratura in letteratura, sviluppato di Melhuish et al. In questo lavoro ogni robot emette un segnale in maniera stocastica. Contando il numero di segnali ricevuti in una finestra di tempo appropriata, i robot sono in grado di stimare la dimensione del gruppo.

Sebbene tale algoritmo garantisca buoni risultati con bassi livelli di errore e alti livelli di stabilità esso richiede tempi di esecuzione piuttosto lunghi. Con lo scopo di accorciare i tempi di esecuzione dell'algoritmo ho sviluppato un secondo metodo di stima basato sulla misura della distanza temporale di due segnali. Misurando il tempo trascorso dalla ricezione di un segnale all'altro ogni robot è in grado di derivare una stima della dimensione del gruppo. Questo algoritmo mostra livelli di errore più elevati ma anche tempi di esecuzione notevolmente ridotti.

Nell'ultimo metodo i robot possono scambiarsi l'informazione sulla quantità di segnali emessi dall'inizio del processo. Ogni robot emette un messaggio e successivamente si limita a aggiornare la propria stima con il contenuto del messaggio ricevuto dagli altri robot. In tale modo ciascuno robot nel gruppo è in grado di derivare una stima precisa delle dimensioni dello sciame in tempi brevi.

Tutti i metodi sono stati studiati e analizzati accuratamente in simulazione per testarne la precisione, i tempi di esecuzione, le proprietà e la capacità di riconoscere mod-

ifiche nelle dimensioni del gruppo. Di ogni metodo sono stati messi in evidenza lati positivi e limiti in modo da dare chiare indicazioni su quale algoritmo risulta migliore a seconda dello scenario di utilizzo.

Acknowledgments

I would like to thank Marco for giving me the possibility to be here at IRIDIA and for all the help he has given me. I would also like to thank Mauro for all his ideas, support and good music.

I would like to thank Carlo, for being an extremely good "boss" and, more important, a very good friend.

Furthermore, I would like to thank my new family: IRIDIA's people. Thank you for all the good time spent together and for making IRIDIA what it is. A special mention goes to Francesco, thanks for the food, the wise suggestions and for being a very good friend and to Colin, thanks for the help with the English (even if you are American) and for making our room the best one in IRIDIA. Thanks also to Ali (mention for the amazing work on our paper), Eliseo, Nithin, Marco, Giovanni, Jeremie, Alex, Matteo, Mattéo, Arne, Antal, Prasanna and all the others.

Thanks also to all my friends here and back in Italy for making me feel at home wherever I am.

Eventually, I would like to dedicate this work to all my family: thanks for being always very supportive and helpful and for giving me the opportunity to be here.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Swarm robotics (Sahin (2005)) is a branch of collective robotics focused on the study of large groups of robots with limited sensor and communication capabilities. Swarm robotic systems naturally display a high level of redundancy and parallelism, thus making them suitable for complex and high risk scenarios such as rescue missions (Casper et al. (2000)) and space exploration (Curtis et al. (2000)). For such applications, some of the robots are likely to be lost or experience failures, thus making the number of surviving robots an important piece of information for tuning the action of the swarm. Moreover, there exist cases in which a specific number of robots gives optimal performance.

In order to tackle failures or to choose the best possible strategy to achieve a goal, robots could use data about the number of the active robots in their group. In this work, we propose three distributed algorithms that let each robot in a swarm estimate, individually and in a reliable manner, the size of the swarm. The first two rely on the minimal assumption that each robot possesses a very basic communication system: each robot can only send signals to its neighbors within a very limited range. The last one relaxes this constraint by considering robots that are able to send and receive a simple message in order to achieve better and faster results.

The first method is based on the pioneering work of Melhuish et al. (1999) on group size estimation, which in turn was inspired by the simple flash synchronization process of fireflies (Smith (1935)). In this work, each robot emits signals in a probabilistic way. By counting the number of signals emitted in a suitable time window, the robots are able to estimate the size of a group. As discussed in more detail in Section 2.3, Melhuish et al.'s method is very promising, but provides a noisy and unreliable estimate of the

swarm's size for practical use. We propose a modification of this method to obtain better results.

As the forthcoming discussion will explain, the main drawback of this method is the long time needed to reach a good estimate. The second method aims at making the estimation time much shorter. To lower the length of the process, we devised a method based on the time intervals between two received signals. By tracking the time elapsed between two signals, each robot is able to derive an estimate of the size of the group. This method displays a higher error level but considerable shorter runtime.

The last method relax the constraint of minimal hardware requirements by allowing robots to exchange simple messages. By exchanging the information about the size of the group, the robots are able to derive an accurate estimate of the group's size in short times.

In Sections 1.1 and 1.2 we give a brief introduction on swarm intelligence and swarm robotics and we describe the motivation behind this work.

In Chapter 2, we present the current state of the art on the topic and give a detailed analysis of the work done by Melhuish et al. We present the Melhuish-Holland-Hoddel (MHH) algorithm with analyses and results.

In Chapter 3, we propose three techniques to obtain group size estimation along with detailed parameters analysis, results and properties. The *message counting (MC)* technique is presented in Section 3.1; the *time measuring (TM)* technique is presented in Section 3.2; the *school trip (ST)* technique is presented in Section 3.3.

Finally, we report conclusions and future work in Chapter 4.

1.1 Swarm Intelligence and Swarm Robotics

Swarm robotics is a fairly new approach to the design and implementation of multi-robot systems. In this kind of systems, a large number of robots interact with each other following behaviors that take inspiration from social insects and swarms of animals.

Nature offers plenty of examples of swarms achieving amazing results through simple rules and interactions: fish schools, bird flocks, ants, bees, wasps, termites are among the most known. Swarms in nature are stunning examples of how order and complexity can emerge from a large number of simple and local interactions.

A well known example of emergence of a global ordered behavior based on simple and local interactions is fish schooling. A school of many thousands of fish is able to

move as a single unit and to react to predators as a whole. Schools of fish proceed gracefully as an harmonious unit, moving to the same direction. Interestingly, when the presence of a predator, however, pushes the school to a sudden change of direction, all its members rapidly respond, almost in unison, moving together as if they where parts of a single organism. While this behavior may suggest some kind of external intelligent and complex coordination mechanism, in fact it arises from simple interactions between individuals.

Schooling results from a balance between attraction and repulsion (Parr (1927)). An individual fish relies on vision and its lateral line for its schooling behavior. Attraction is controlled mainly by vision – when a fish goes too far from its neighbors it moves closer to nearby fish. With the lateral line, instead, a fish senses if it is too close to a neighbor and must move away to avoid a collision. Schooling emerges just simply from these two basic behaviors.

The same concept, complex behaviors emerging from simple rules, is behind many other natural behavior and is called *"self-organization"*. A very good definition of self-organization can be found in Camazine et al. (2003):

> Self-organization is a process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying interactions among the system's components are executed using only local information, without reference to the global pattern.

Self-organized behaviors are successful in nature because they display several interesting properties. First of all they are *robust*. Robustness means that the swarm (or the group) is able to survive even when some or several individuals die or are incapacitated. Ant colonies can survive even if many of their workers are killed and bees have brought this characteristic to the extreme: they are ready to sacrifice some individuals in order to protect their hive. These systems are robust because they are *distributed* and they rely only on local communication. They are distributed in the sense that there is no leader (queen bees or ants do not participate during everyday activities). Moreover, all the members of the swarm are equal and have the same behavior and so are easily replaceable. Other interesting aspects are *adaptivity* and *cooperation*. Swarms are able to cope with variations in the environment (such as modifications in the landscape or changing in the availability of resources) and use their number to tackle obstacles that are otherwise impossible to tackle by a single individual. Ants are able to transport ob-

jects that are several times heavier than the maximum strength of a single individual by doing it cooperatively (Figure 1.1(a)) or are able to build "living bridges" linking several ants together in order to be able to overcome a gap that is too large for a single ant (Figure 1.1(b)). Moreover, a group of individuals is able to cover an area much larger than a single one. Bees are able to discover food sources in areas relatively distant from their nest and communicate the discovery to other members of the hive. Examples of these amazing behaviors are shown in Figure 2.4.

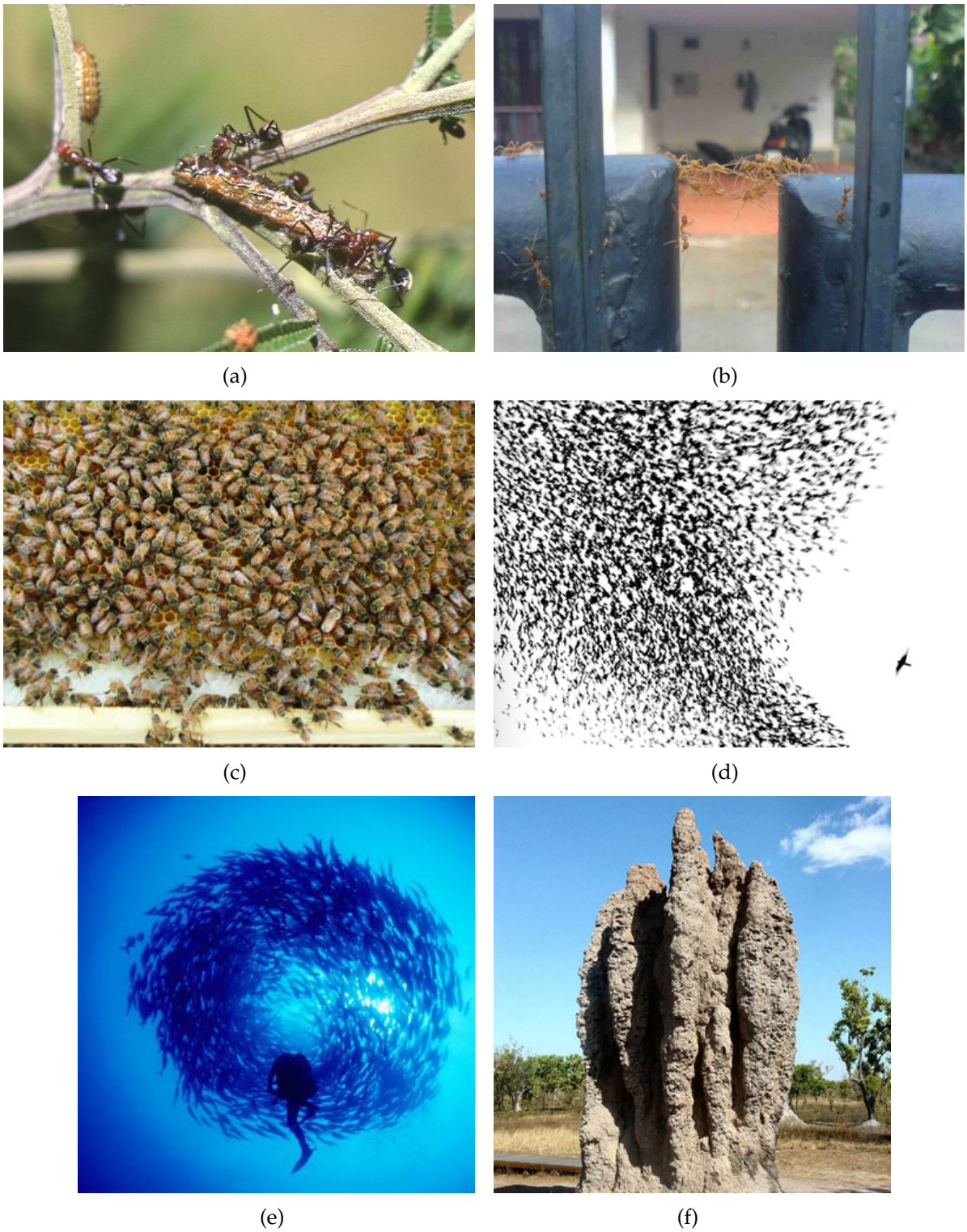
We would like to be able to mimic results obtained by swarms of social animals. Until now, creating powerful and versatile robots has proved to be a very hard task both on the hardware and control aspect and, while many efforts are going in this direction, a new field is trying to explore another paradigm. Swarm robotics, as a research field, is trying to build systems composed by simple robots cooperating to reach a common goal, applying bioinspired ideas to computer science. This fairly new approach is the application of swarm intelligence principles to robotic systems (Dorigo and Birattari (2007); Bonabeau et al. (1999)).

It is not easy to give a good definition neither of swarm robotics nor of swarm intelligence (a very good analysis is conducted by Beni (2005)). Swarm intelligence systems are difficult to define mainly due to the difficulty of defining "intelligence". On the other hand, a working definition of swarm robotics was given by Dorigo and Sahin (2004):

> Swarm robotics can be loosely defined as the study of how collectively intelligent behaviors can emerge from local interactions of a large number of relatively simple physically embodied agents. Swarm robotics studies are often inspired by the observation of social insects - ants, termites, wasps and bees - which stand as fascinating examples of how collectively intelligent systems can be generated from a large number of simple individuals.

They also identify four criteria to distinguish this field from other multi-robot systems: *large numbers of robots*, *homogeneity*, *cooperation* and *locality*. Swarm robotics systems are characterized by large groups of simple robots (tens, hundreds, thousands) composed by one (homogeneous swarms) or few kinds (heterogeneous swarms) of equivalent robots that must cooperate through local interaction and communication to achieve a task.

Up to now, the large majority of experiments done in swarm robotics dealt with small groups of robots, usually less than 30 - 40. This does not mean that these systems cannot be considered swarm robotics. To cite again Beni (2005):



(a) (b)

(c) (d)

(e) (f)

Figure 1.1: Natural example of swarms. (a) a group of ants performing collective transport (copyright by earth-info-net.blogspot.com) (b) ants building a living bridge to cross a gap (copyright by thinkorthwim.com) (c) a swarm of bees on a hive (copyright by beecare.com) (d) a flock of birds escaping a predator (copyright by pbs.org) (e) a school of fish swimming in circle over a diver (copyright by reefrants.blogspot.com) (f) a termite nest (copyright by a-z-animals.com).

> The use of labels such as swarm robotics or collective robotics / distributed robotics should not be in principle a function of the number of units used in the system. The principles underlying the multi-robot system coordination are the essential factor. The control architectures relevant to swarms are scalable, from a few units to thousands or million of units, since they base their coordination on local interactions and self-organization. The fact that only small groups of robots have been presented in most of the swarm robotics literature is a side effect of cost of robotic equipment and of the number of technologies involved to make robots working. Making a single mobile, autonomous robot working in a reliable way is already a big challenge nowadays, and even more so for a robotic swarm.

The use of simple robots instead of more powerful and complex ones has many interesting advantages today as well as in the future, when more complex, powerful and cheaper robots will be available. This kind of minimalism is a very good aspect in the sense that the less there is, the less there is to go wrong; this applies both to hardware and to software. Simpler robots means less hardware failures while simple software should mean less bugs. If things are simple they are easier to control and to manage and in the long run this means more reliability.

1.2 Group Size Estimation: Motivation

In swarm robotic systems, groups of tens or hundreds of robots cooperate to achieve a goal, but big numbers are not always a good thing. Let us consider an example. We want to clean the floor of a very large room by minimizing the time needed to complete the task. A single man needs too long time and too much effort to do it. If more cleaners are assigned to the task, the time needed to complete it will be reduced and the effort done by the single will be less, but this holds up to a certain point. If the number of people involved in the task is too high, they will interfere with each others, stepping on areas of the floor already cleaned or bumping into each other. Moreover, if the resources needed to clean the room (e.g. mops, cleaning solutions) are limited and shared, the cleaners will compete over them and thus the overall performance will be furtherly reduced. Hence, it is possible to identify an ideal group size for this specific task, according to several factors such as the size of the room, the availability of the cleansing products or the ability of the single cleaner.

The same situation applies to swarm robotics. A minimum number of robots is often required for specific tasks (e.g. cooperative transport) and as the number of robots assisting with the task increases, the overall performance increases superlinearly (Mondada et al. (2005)). Beyond a certain threshold, the negative effects caused by interference and competition make the overall performance decreases (Lerman and Galstyan (2002)). Furthermore, coping with the increase of interference to fight performance loss obliges the designer to study more complex solutions. Therefore, we can identify an optimal number of robots corresponding to maximum performance.

Keeping track of how many robots are composing a group is very useful if the objective is to divide the set of the robots into smaller groups to better tackle a task. Based on the size of the group, robots may switch to the fittest operational regime and/or divide into groups of optimal sizes for the tasks to perform.

Moreover, keeping track of the size of the group is fundamental in failure management. In the typical usage scenario, a swarm of known size is deployed and failures are expected as the task progresses, thus lowering the number of active robots in the group. When the loss of robots drives the swarm's size below a certain threshold, robots may trigger self-rearing procedures or send messages to the operator asking for the addition of new robots. Another scenario could be the adoption of different strategies according to the estimated size of group. For instance, in a risky scenario if the robots have the information that the group is very large, they can choose to use a faster but riskier strategy knowing that even if they suffer many losses they will still be able to complete the task. On the other hand, if they know that the group is small, they can choose to adopt a slower but safer strategy in order to have better chances to complete the mission.

Chapter 2

Group Size Estimation

2.1 Existing Systems

Up to now, to the best of our knowledge, there are no studies that explicitly concentrate on estimating the size of a group of robots. While a number of studies have been done on aggregation and group size regulation (e.g. Dorigo et al. (2004); Martinoli and Gambardella (1999)), none of these works presents a method to explicit estimate the group's size. In group size regulation and aggregation the task is typically achieved without an explicit knowledge of the size of the group. In the usual scenario, groups that are a fraction of the total swarm are formed. This means for example forming three groups in a 30%-60%-10% ratio and this is usually done without knowing how many robots are present in the swarm and in the formed groups. In general the size of the group is considered to be known *a priori* and modifications (e.g. due to failures) are supervised by the human researcher normally by ending the experiment and fixing the failing robots.

With th aim of devising a group size algorithm, we can differentiate between three kinds of approaches: centralized, semi-centralized and distributed.

In *centralized* approaches, the estimation is done by an external single entity. This entity is able to estimate (often explicitly counting) the number of robots in a group. An example of this is a camera overseeing an arena in which an experiment is ongoing. A central computer, using the information extracted with the camera (i.e. extracting the number of colored blobs in the image), can count the number of robots and thus gather an estimate of the size of the group. Another way to count robots is using explicit communication. Every robot sends its ID to a central computer that in this way will have the knowledge of every robot composing the group and thus its size.

These systems, while being generally quite accurate, rely on very strong assumptions: global knowledge or very powerful and complex communication capabilities which, in general, are not available in real swarm robotic applications. Moreover centralized methods suffer from two very important issues: single point of failure and poor scalability. Single point of failure means that if a problem with the device (i.e. the camera or the computer) occurs, the whole system stops working. Poor scalability means that in centralized systems, performance drops as group size increases. This is caused by the fact that every piece of information needs to be sent to a central unit and be processed by it so as the number of robots increases, the number of messages increases as well, while the central unit is able to manage only a limited number of data at the same time. Hence, centralized methods are in general not indicated for scenarios involving large swarms of robots.

Another option are semi-centralized solutions. With *semi-centralized* systems, we refer to those cases in which a subset of robots, or even a single individual, has the special role of estimating the group's size. These robots can be physically different from the others or just running a different controller. In O'Grady et al. (2009) and Pinciroli et al. (2009) a swarm of heterogeneous robots is used to achieve group size regulation. A flying robot called *eye-bot* is able to distinguish the number of robots that are in its visual area by counting the number of active LEDs. Knowing how many LEDs each robot possesses, the robot is able to count the number of individuals below it. This strategy is very effective in the process of task allocation as presented in the paper, but if we wanted to port this strategy to perform explicit group size estimation we would observe some drawbacks. First of eye-bots cover only a part of the arena: the system is able to count only the robots under the radius of the eye-bots. This means that we would need a high number of robots to cover a large area. Furthermore, failures of single robots would reduce the coverage, thus making the system not scalable or robust. In addition, without modification in the system, these "observers" would only be able to count how many robots are present under them in a particular instant, not the global size of the group. The information is not integrated over time and is not shared between robots. To conclude, while this method is well suited for the task allocation problem addressed in the work, it would require substantial modifications to be suitable for group size estimation.

If centralized systems are not really diffused in literature, the *distributed* ones are even less common. In McLurkin and Yamins (2005) an algorithm similar to the link-state

routing protocol is used. Each robot broadcasts its ID and the list of the IDs received in the previous steps along with a time-stamp. Each robot then merges the received list with its own, according to the time-stamp, and repeats the broadcast. After some steps, the algorithm converges and all robots have a complete list of all the members of the group and thus the size of the group. This process, while being quite fast, requires a large amount of inter-robot communication and the message itself is quite complex. Robots like the one we are considering are not able to have this kind of intensive and complex communication. Furthermore, a solution like this scales very poorly because of the increasing amount of messages traveling in the group and the increasing memory and computing power needings.

Melhuish et al. (1999) have proposed a method to achieve group formation using an algorithm able to roughly estimate the group's size. Even though the aim of their work is not to explicitly compute the size of the group of robots, they proposed a way to estimate it in a simple and distributed way with minimal communication requirements. This method is the base for our algorithm and is explained in detail in Section 2.3.

2.2 Methodology

2.2.1 Quality Assessment

The quality of the obtained estimate is an important factor in choosing what and when to use a particular kind of estimate. To better evaluate and compare the different algorithms discussed in this work, we hereby state three properties that a method should have to be usable in real world scenarios.

The first desirable property such a method should display is low individual estimation error, i.e., the difference between a robot's estimate and the actual group size should be small. If the error is too big, the estimate is unreliable and so it cannot be used in practice. On the contrary, an overall low error is necessary for every decision process relying on this value. It must be noticed that, given that the estimate can change at every time step, the considered error should be an average over time. For this reason we define the mean relative error of robot i as:

$$E^i = \frac{1}{t_2 - t_1} \sum_{t=t_1}^{t_2} \left| \frac{N - \hat{n}_t^i}{N} \right|$$

where N is the actual number of robots in the swarm, $\hat{n}_t^i$ is robot i's t-th estimate and t_1 and t_2 define the time window during which the estimates are averaged.

Moreover, due to the distributed nature of the algorithm, the estimate can be different from robot to robot. A way to measure the global error is the average of the error of every robot:

$$E = \frac{1}{N} \sum_{i=1}^{N} E^i.$$

Secondly, each robot's estimation should be *stable*, meaning that once settled on a value, the estimate should not fluctuate significantly. A system in which the estimated value is fluctuating strongly around the correct value can have a low error but it will be not usable.

Finally, another important property is a high degree of *agreement* throughout the swarm, i.e., the estimates of the individuals in the swarm should not be too different among each other. Should each individual have a totally different estimate, then every robot would react to this information in a different way posing potential coordination issues. On the other hand, knowing that every robot of the swarm has the same estimate ensures that the robots behave in the same way to the same situation. While stability deals with the average over time of the error value, this property deals with the average done over the robots. Let us consider a group in which half of the robots have an estimate of 0 while the other half has an estimate of $2N$. On average, the estimate would be correct, so the error would be zero, but of course such output for the individual robots would be wrong.

All our results have been obtained through extensive simulations. Implementation on real robots[1] is currently under development.

2.2.2 The ARGoS Simulator

All the results presented in this work are obtained by using the ARGoS simulator (Pinciroli (2006)). This simulator was developed in the framework of the Swarmanoid[2] project to have a flexible yet powerful tool. This simulator is used to test the controllers in order to have preliminary results of the algorithm and shorten developing times. Moreover the simulator enables us to have reproducible and thus analyzable results.

[1]The platform we are porting our software to is the *e-puck* robot (`http://www.e-puck.org`).
[2]For further information go to `http://www.swarmanoid.org`

ARGoS provides several different physics engines: 2D kynematics, 2D dynamics, and 3D dynamics. ARGoS can run more than one physics engine in the same experiment and the user can choose which robots to simulate with which engine so that simulation is optimised and accurate. Simulation performance is a critical issue, made even more critical by the fact that we want to be able to evolve controllers.

The modular architecture of ARGoS allows the user to easily add new actuators, sensors, physics engines and renderers. The simulator can be configured with an intuitive XML file and new modules programmed by the user are automatically registered and made available for use in the XML file.

Finally, ARGoS is designed to make the transition between simulated robots and real robots seamless. The control interface to write robot controllers is common to both simulated and real robots. Thus, a controller developed for a simulated robot needs only to be recompiled before it can be run on the real robot platform.

Some screenshot of the simulator at work are shown in Figure 2.1.

2.2.3 Robot Deployment

The proposed methods work as long as the robots form a connected graph in which robots are nodes and communication links are graph arcs. This means that the robots are able to estimate the size of the group both in static conditions (robots not moving) and dynamic conditions (robots moving). We tested the methods in several different scenarios and the performance was not affected by the movement of the robots. Of course, problems like delays in the transmission of the messages could affect the quality of some measurement and ensuring that the robots are always connected is sometimes not trivial. Moreover, particular shapes, like lines of robots, bring to increased times for the spreading of messages. To reduce the incidence of these problems in our simulations we deploy our robot using a square disposition. While this deployment was arranged artificially, there are methods to achieve this result in swarm robotics. As an example we can refer to the work by Spears et al. (2004). This is a good way to ensure low latency in the diffusion of the signals and to assure connection between robots. A picture of the robots in their disposition is shown in Figure 2.2.

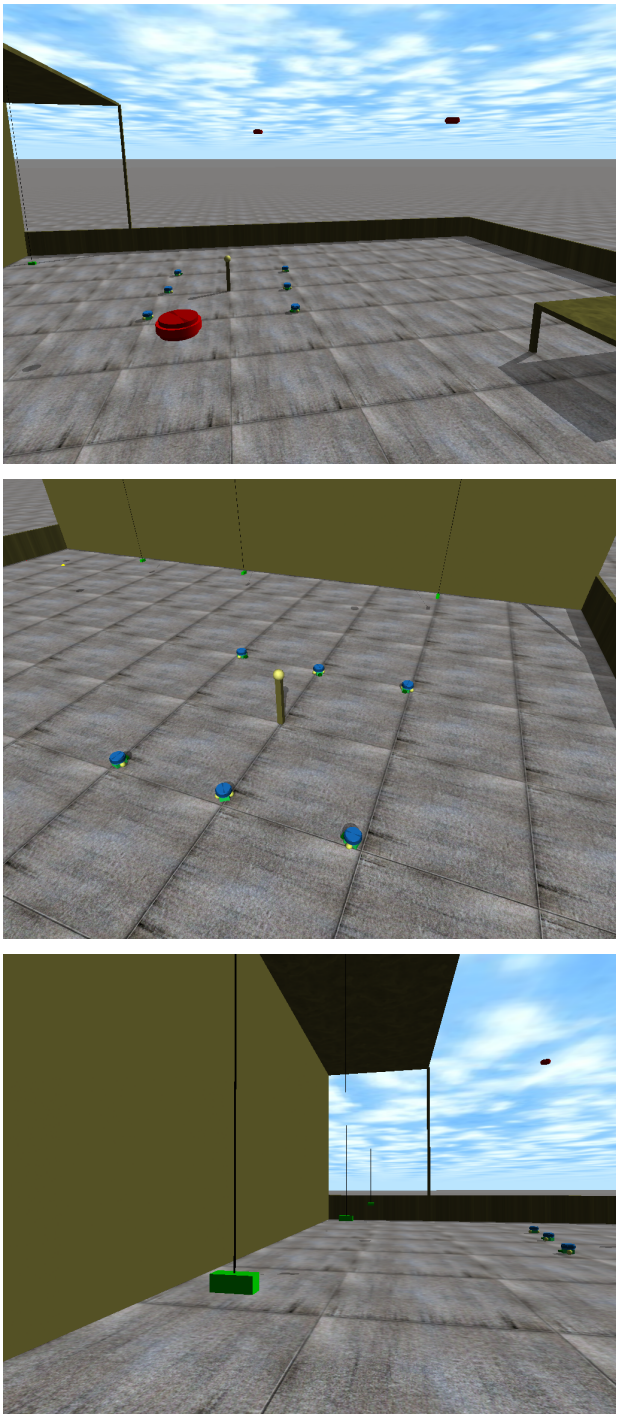


Figure 2.1: Some screenshots of the ARGoS simulator.

2.2.4 Characteristics

In the design of a group size estimation mechanism for swarms of robots, the most desirable features are scalability, robustness and distributedness. Scalability means that performance (in our case the estimation error) does not significantly drop as the group size decreases. Estimation techniques should also be robust: methods that rely on the assumption that all the individuals are always working are, first of all, not realistic. In addition, group size estimation is a good method to keep track of the percentage of working / non-working robots and thus it needs to be robust to single robots failures and should be able to keep track of the new group's size. Lastly, it has to be distributed, otherwise it will lack the characteristics of robustness and scalability we have discussed so far. A method relying on the presence of a leader will fail completely if the leader fails (non robust) and will behave poorly in very large groups (non scalable).

2.3 The Melhuish-Holland-Hoddel Method

Melhuish et al. (in Melhuish et al. (1999); Holland and Melhuish (1997); Holland et al. (1997)) proposed a method that allows the robots in a group to have an estimate of the size of the group in a distributed, robust and simple way. Hardware requirements are very limited as the robots need only the ability to emit and receive signals. This can be done through visual signals (LEDs and camera), infrared communication or any other means. In our simulations we used a simulated range and bearing board (a real implementation has been performed by Álvaro Gutiérrez et al. (2009)) for the e-puck robot. This board allows the e-pucks to emit and receive 16-bit messages. Furthermore, upon receipt of a message, the board allows to estimate the distance and angle of the sender.

The robots exploit a mechanism similar to those used by fireflies to achieve synchronization. The Melhuish-Holland-Hoddel method (MHH) is a modification of Mirollo and Strogatz' model in which fireflies behave like coupled oscillators (Mirollo and Strogatz (1990)). A more detailed view of fireflies synchronization mechanism is given in Section 2.3.1, while an analysis of the algorithm proposed by Melhuish et al. to estimate the group size is presented in Section 2.3.2.

2.3.1 Inspiration

Fireflies are beetles (family *Lampyridae*) that are a familiar sight on warm summer evenings in most of the world. Their are known for their flashing abdominal lanterns, which are used to attract mates or to prey. Different species of fireflies have developed different patterns of mating signals. One particular form of luminescent behavior can be seen from India east to the Philippines and New Guinea. This particular kind of behavior has fascinated explorers and naturalists for hundreds of years. In these regions, enormous aggregations of fireflies gather in trees and flash in near-perfect synchronization (Figure 2.3).

Smith (1935) wrote:

> Imagine a tree thirty-five to forty feet high, apparently with a firefly on every leaf, and all the fireflies flashing in perfect unison at a rate of about three times in two seconds, the tree being in complete darkness between flashes. Imagine a tenth of a mile of river front with an unbroken line of mangrove trees with fireflies on every leaf flashing in synchronization, the insects on the trees at the ends of the line acting in perfect unison with those between. Then, if one's imagination is sufficiently vivid, he may form some conception of this amazing spectacle.

This mechanism has been studied for a long period and while the *why* remains uncertain, the *how* is now known. This mechanism is thought to be related to reproductive behavior. Synchronized flashing is performed entirely by males that maintain position spaced on individual leaves. Females fly to these trees to mate with males. The females emit irregularly timed, longer flashes that are dimmer than the male's. The males will start to flash soon after sunset, while synchronization builds up slowly through the night.

Synchronization among males could serve at least three purposes. First, synchronization could accentuate the male rhythm. If females select males based upon a species-specific flash repetition rate, then the chaotic flashing of dense aggregations of males may thwart the female's ability to assess the males' flashing pattern. By mutually synchronizing their rhythm, each male benefits by allowing its flashing rate to be evaluated by a potential mate. Second, a similar argument based on enhanced detection applies if mating requires that males visually detect females. If a portion of the mating sequence requires that males detect the flash of nearby females, synchronization could serve as a

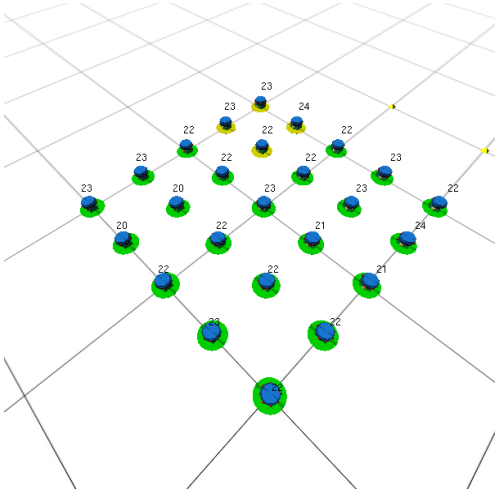


Figure 2.2: The disposition of the robots in our experiments.

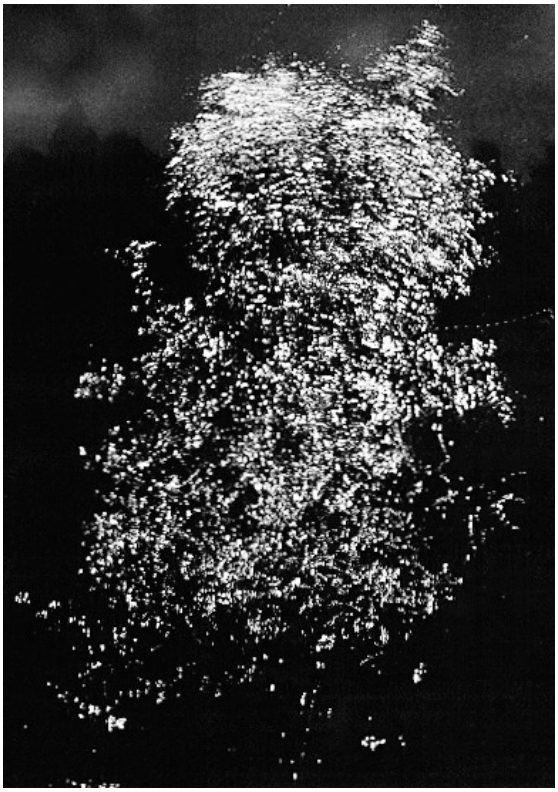
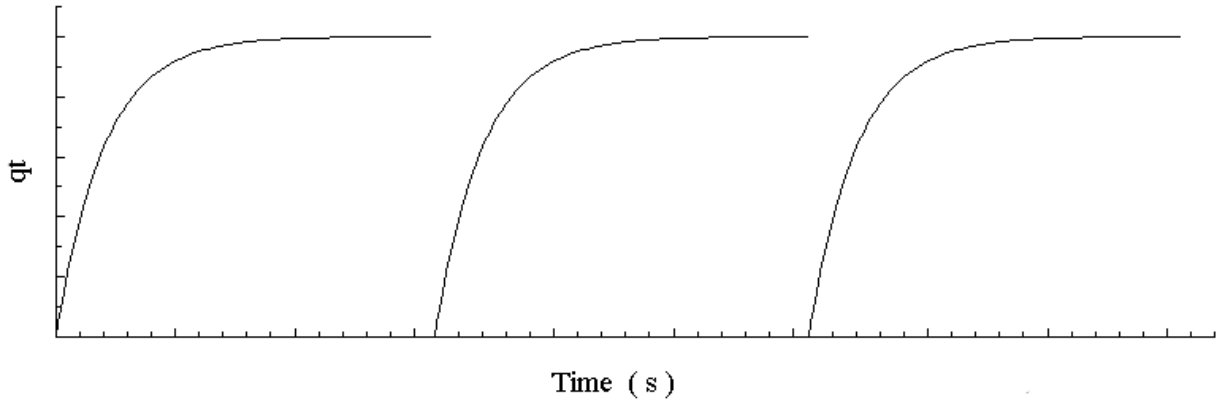


Figure 2.3: A tree with synchronous flashing fireflies (copyright by montefiore.ulg.ac.be).

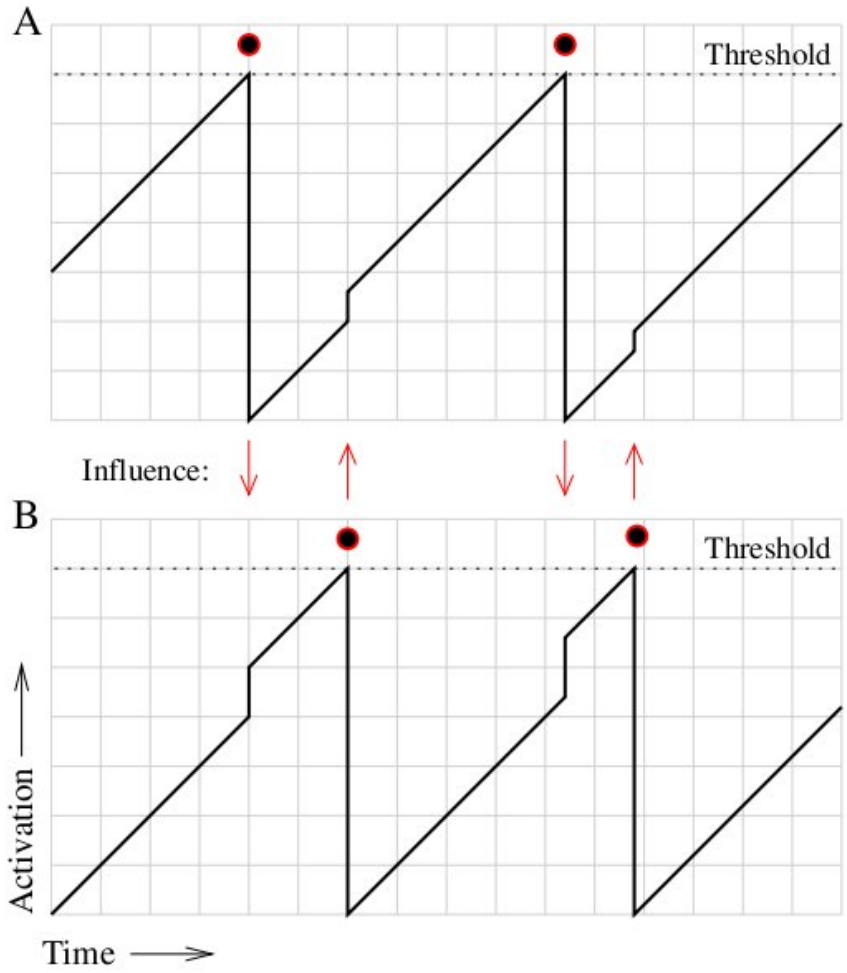
noise-reduction mechanism allowing males better to find females in the dark interflash periods. Third, synchronization could be a signal enhancement mechanism enabling small groups of synchronizing males to attract larger number of females.

The biological mechanisms behind this kind of synchronization were understood by various scientists around 1960 and were mathematically described by Mirollo and Strogatz (1990). Case and Buck (1963) and Buonomici and Magni (1967) found that each flash is triggered by nerve impulses in the brain. Experiments have supported the role of the brain as a central timer (Bagnoli et al. (1976)). An isolated male firefly, thanks to the natural oscillator in its brain, continues to flash with a constant period (Figure 2.4(a)). Understanding what happens when several fireflies are together and how synchronization arises is a matter of understanding the interactions between individuals. A key feature of the system is that the pattern emerges as a result of multiple interactions among the fireflies. Synchronization is not imposed by any influence outside the system, such as leaders, supervisors or external physical cues. Synchronization arises from within, based on local interactions among fireflies that follow the simple rule: a neighbor's light emission shifts the timing of one's own light emission. When a firefly perceives a flash, it will increase its "internal counter" by a certain value shifting its phase forward and thus moving it towards the same phase of the signaling firefly. This of course will bring every fireflies to the same phase, and given that the period is common this will produce synchronization. A graphical explanation can be seen in Figure 2.4(b). The two graphs show two different fireflies interacting. The big dots are the moment in which a firefly emit a signal. As proved by Mirollo and Strogatz, the phase of the two individuals moves towards synchronization.

One of the most important features of this system is its ability to work as long as there is visual interaction (i.e. communication) between synchronizing fireflies. This means that as long as fireflies can be represented as a connected graph, in which fireflies are nodes and visual interaction are links, the system is always able to achieve synchronization. This is quite obvious as, if two groups of fireflies are not able to perceive each other, they will not be able to synchronize themselves. On the other hand, this significantly lowers the necessary communication capabilities of the fireflies. It is sufficient that an individual sees just another individual, i.e., the system works even when the individuals have a very limited range.

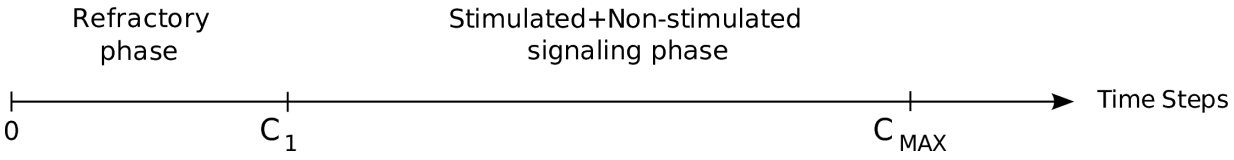


(a)

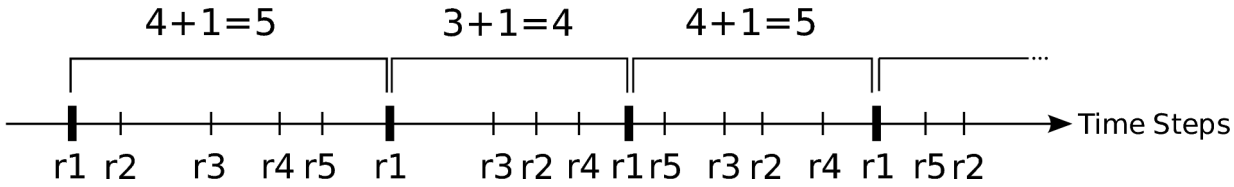


(b)

Figure 2.4: (a) Model of the internal timer of a firefly (b) Model of the interaction between two fireflies (Courtesy of A. L. Christensen).



(a) The two-phase cycle proposed by Melhuish et al.



(b) An example of the counting process.

Figure 2.5: The MHH method.

2.3.2 Algorithm

Fireflies do not use their synchronization process to count themselves. Nevertheless, this basic idea was taken as an inspiration by Melhuish et al. for their work. The MHH method is based on a simple idea: if a robot emits a signal with a certain probability, the more robot there are, the more signals will be emitted. In other words, if in a group of N individuals, an individual emits a signal with a fixed probability p, the size of the group can be obtained from the frequency of signals emitted by the group. Hence, if we were able to determine the time elapsed between two signals, we could determine the size of the group.

Instead of considering the time delay between two signals, it is possible to count the number of signals in a suitably defined amount of time. In this way, it is possible to obtain a more stable and accurate estimate. The MHH method follows this approach.

Each robot possesses an internal counter c ranging from 0 to C_{MAX} which is increased at each time step by a fixed quantity $\delta = 1$. When $c > C_{MAX}$ the robot emits a signal and sets $c = 0$. Depending on the value of c, a robot can be in one of the two phases of the algorithm: it is either in the *refractory* phase when $c \in [0, C_1]$ or in the *stimulated+non-stimulated* signaling phase when $c \in (C_1, C_{MAX}]$ (see Figure 2.5(a)).

During the *stimulated+non-stimulated* signaling phase, at each time step a robot can emit a signal with probability p. When this happens, the robot has emitted a *non-stimulated* signal. After signaling, the robot resets c to 0. A neighboring robot that perceives the signal resets c to 0 and emits a signal too. In the latter case, the robot has emitted a *stimulated* signal. This causes a cascade of stimulated signals across the

swarm that are crucial to communicate the information throughout the whole swarm. As it is possible to see in Figure 2.6, even if a single robot is able to perceive only the signals from its nearest neighbors, a signal emitted by one robot reaches all the others. This works thanks to a "relaying" mechanism: a robot that receives a signal emits a similar signal in response. In turn, this second signal can be perceived by an individual that is able to see the second robot but not the first. Of course, the third signal will be received by a forth robot and so on.

To avoid an infinite sequence of signal waves traversing the swarm back and forth, robots that have signaled enter into the *refractory* phase which prevents them from signaling even if a neighbor does. Therefore, robots cycle between two successive phases: the *refractory* phase, in which a robot ignores its neighbors' signals, and the *stimulated+non-stimulated* signaling phase, in which a robot can signal or be stimulated to signal. The pseudo-code of this algorithm can be found in the Appendix.

With this method, in a swarm of N robots, a robot emits on average a non-stimulated signal every $N-1$ stimulated signals. Each robot counts the number s_t^i of stimulated signals it has emitted since the last non-stimulated one. Therefore, $\hat{n}_t^i = s_t^i + 1$ can be used as an estimate of the group's size (see Figure 2.5(b)). This method works on the assumption presented before: given that every robot has the same probability p to emit a signal, we expect to observe a signal from a specific robot every N signals. This means that the robots can estimate the size of the group according to how many signals they receive between two non-stimulated signals.

2.3.3 Discussion

Melhuish et al.'s method displays dramatic fluctuations of the values of s_t^i over time.

This is the key drawback of the method, because it entails two undesired consequences. First, for an individual robot it is not possible to ultimately decide about the size of the group, thus making the estimate basically useless. Second, the swarm as a whole displays a very low degree of agreement – even averaging an individual's estimate with its neighbors' is not likely to improve the method's performance sensibly. In their method, Melhuish et al. dampen fluctuations with a weighted average of $\hat{n}_t^i$ over time:

$$\hat{n}_t^i = \alpha(s_t^i + 1) + (1-\alpha)\hat{n}_{t-1}^i.$$

In their paper, Melhuish et al. suggest $\alpha = 0.85$. Figure 2.7(a) shows a run with optimal

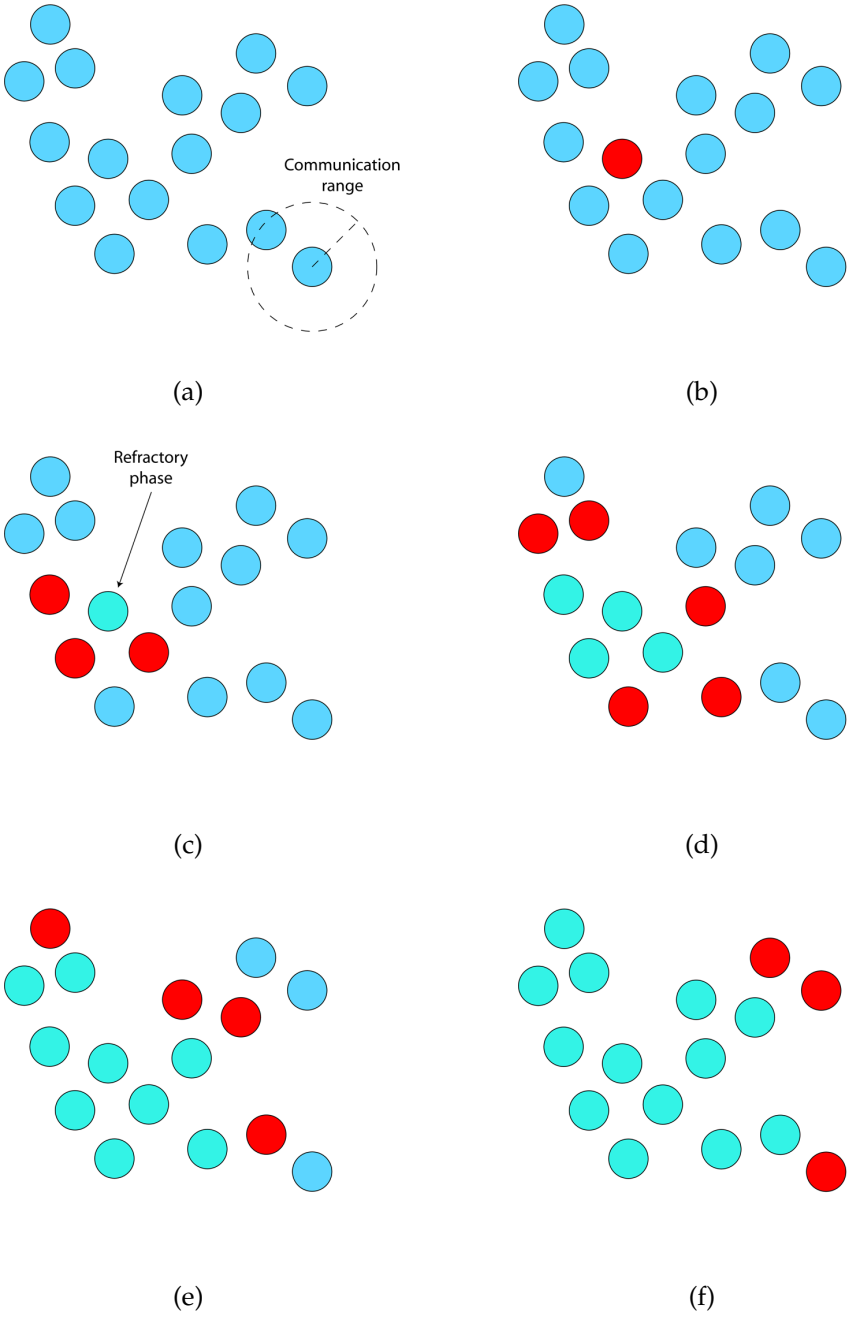
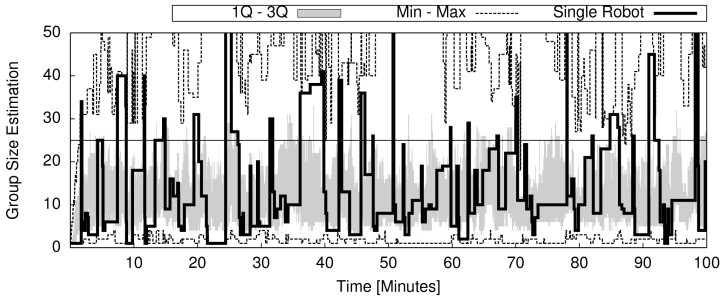
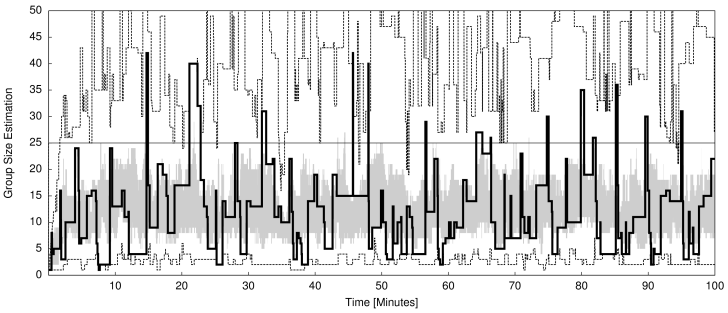


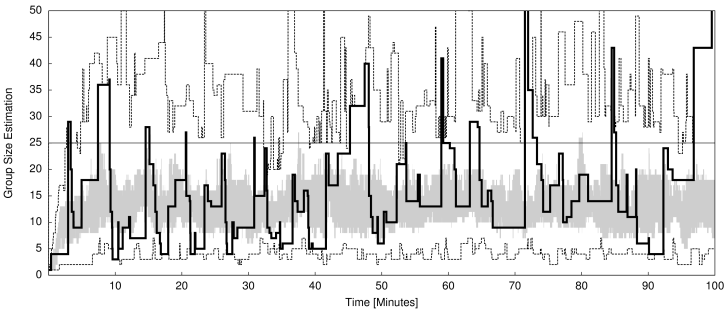
Figure 2.6: A graphical representation of the transmission of signals. Figure 2.6(a) shows the range of a single robot. The other images show how a signal is relayed across the swarm.



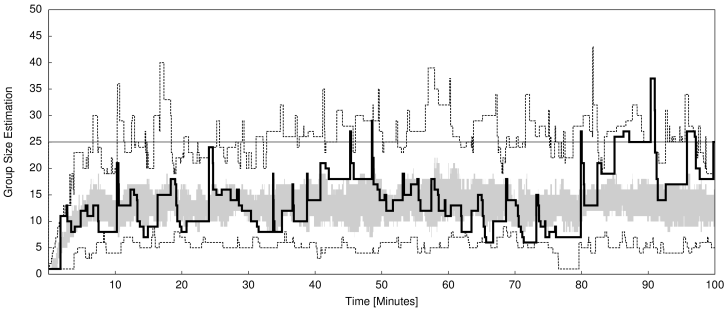
(a) $\alpha = 0.85$



(b) $\alpha = 0.65$



(c) $\alpha = 0.45$



(d) $\alpha = 0.25$

Figure 2.7: Results of test runs of the MHH method with 25 simulated robots. The graphs show the distribution of the estimates of the group's size for each robot over time. The thin dotted line delimits the maximum and minimum estimates in the swarm; the gray area corresponds to the estimations falling into the first and third quartiles; the black solid line is an example of estimate of a single robot. Various values for α are considered.

α	C_1	C_{MAX}	p	*Error [%]*
0.25	20	260	0.01	44.66
0.45	20	260	0.01	42.25
0.65	20	260	0.01	42.66
0.85	20	260	0.01	58.24

Table 2.1: Mean relative error of the MHH method with different α values.

values found through grid search. The estimate has an elevate overall error (58.24%) and little to none agreement and stability. Lowering the value of α can help in dampening the fluctuations but the overall error does not improve significantly. Results can be seen in Figure 2.7 and in Table 2.1.

In Figure 2.7 , we adopt the following conventions: the thin dotted line delimits the maximum and minimum estimates in the swarm; the gray area corresponds to the estimations falling into the first and third quartiles while the black solid line is an example of estimate of a single robot. The solid line is provided as a reference to understand the behavior of a single, random chosen, individual. A qualitative measure of the three properties is provided by these graphs. The distance of the lines from the reference line expresses the error; the thickness of the max-min span or of the quartile span measure the level of agreement across the swarm; the oscillation of the lines indicates the stability of the result.

We have done a complete analysis of all the parameters composing the system to better understand the reason behind these high error levels. The signaling probability p plays a key role in shaping such fluctuations: for a high value of p, robots signal too frequently in a non-stimulated manner. In this way, signals are likely to overlap, thus making the estimates $\hat{n}_t^i$ significantly lower than N. For this reason, small values of p are preferable.

Our experiments showed that the length of the *refractory* phase C_1 has no effect the error of the system, but affects its stabilization speed. High values for C_1, in fact, oblige the robots to wait for a long time before exiting the refractory period, thus slowing down the entire process. In our experiments we set the length of this phase to the same value used in Melhuish et al.'s experiments: $C_1 = 20$.

Finally, the choice of a value for C_{MAX} derives from a trade off between scalability and speed. In fact, if C_{MAX} is too low, there could be not enough time for all the robots to signal, thus making the method not scalable with N. The value of C_{MAX} depends also on the probability of emitting a non-stimulated signal p, so that a high C_{MAX} value

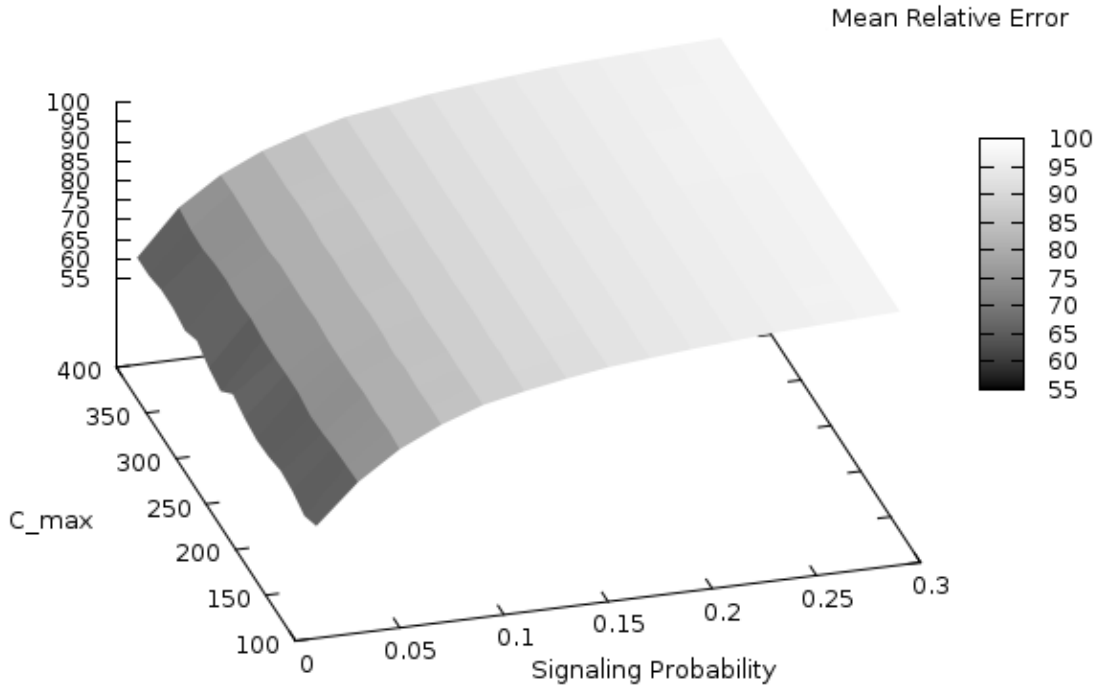


Figure 2.8: Graph of the mean relative error related to parameters. The mean error strongly decrease as the signaling probability decreases while it is less sensitive to C_{MAX} variations.

with a low probability p increases the time needed by the robots to obtain the estimate of the group's size.

A comparison of all the tested parameters is shown in Figure 2.8. The graph shows that the mean relative error increases as p increases, while it is inversely proportional to C_{MAX}. Nevertheless, the mean relative error remains high even with optimal parameters.

The reason behind the poor performance of the algorithm is that it is unable to impose a global order in the emission of non-stimulated signals. When a robot emits a non-stimulated signal, after the *refractory* phase, it has the same probability as any other robot to emit another non-stimulated signal, so no order is imposed on the system. This ultimately prevents the robots to achieve a good estimate of the group's size. Let us consider the series $S(k)$ of the IDs of the robots that (over time) emit a non-stimulated signal (i.e., $S(k) = i$ if the swarm's k-th non-stimulated signal has been emitted by robot

i). In the ideal scenario a robot emits a non-stimulated signal exactly every N stimulated signals, thus the order in the signals should be perfect. We should observe always the same sequence of signaling robots every N signals. Table 2.2 presents the IDs (from 0 to 24) of the robots when they emit a non-stimulated signal. The output of the algorithm is a continuous list but here is divided in columns of length 25 to better appreciate the presence or lack of order. We report an excerpt of the results of an experiment in Table. It is clear that there is no order in the signals and, more important, between two signals of a particular robot we do not observe the presence of the signals of the other individuals. Instead, it happens quite often that some robots (e.g. robot with ID 3) signal very shortly after the previous signal while others (e.g. robot with ID 22) are silent for long periods.

In order to solve the problems related to this method, we propose a modification to obtain a more stable and accurate estimate in Chapter 3.

	Cycles				
IDs of the robots emitting a non-stimulated signal	3	3	3	3	3
	16	16	16	16	16
	9	9	9	9	9
	10	10	10	10	10
	24	24	24	24	24
	19	19	19	19	19
	23	23	23	23	23
	15	15	15	15	15
	5	5	5	5	5
	1	1	1	1	1
	22	22	22	22	22
	4	4	4	4	4
	17	17	17	17	17
	8	8	8	8	8
	12	12	12	12	12
	20	20	20	20	20
	6	6	6	6	6
	7	7	7	7	7
	18	18	18	18	18
	0	0	0	0	0
	21	21	21	21	21
	14	14	14	14	14
	11	11	11	11	11
	2	2	2	2	2
	13	13	13	13	13

Table 2.2: Ideal sequence of the IDs of the robots emitting a non-stimulated signal in a group of 25 individuals. Data is divided into columns of length 25 to better appreciate the periodicity.

	Cycles				
IDs of the robots emitting a non-stimulated signal	3	21	24	16	9
	21	21	4	23	8
	18	11	12	13	1
	10	13	17	9	1
	18	7	6	24	20
	5	4	21	17	2
	9	1	10	6	8
	3	8	11	19	16
	17	11	10	1	24
	3	12	18	16	5
	4	5	2	19	6
	8	21	9	14	2
	5	7	4	15	23
	16	5	8	24	2
	2	23	15	16	3
	1	11	17	4	23
	17	1	0	14	10
	3	5	15	21	23
	9	20	7	4	13
	0	11	14	18	13
	9	7	7	13	4
	0	17	5	18	6
	7	21	9	7	19
	9	18	13	12	5
	0	5	14	22	20

Table 2.3: Sequence of the IDs of the robots emitting a non-stimulated signal for the MHH method with 25 robots. Data is divided into columns of length 25 to better appreciate the lack of periodicity.

Chapter 3

Three Algorithms for Group Size Estimation

In this chapter we present three algorithms for group size estimation. Each algorithm is based on distributed principles and local communication to ensure robustness and scalability. The first two methods have minimal communication requirements: the robots only need to be able to emit and receive a signal from the nearest neighbors. The last method relaxes this constraint: the robots communicate by sending a simple message, i.e. an integer value.

The first method is an improvement of Melhuish et al.'s method. The second algorithm is based on the dual principle: instead of counting the number of message received in a suitable amount of time, each robot tracks the time elapsed between two signals and thus derives an estimate of the size of the group. In the last algorithm, each robot emits only one message, communicating the number of the received messages. The number emitted by the last robot is considered to be the size of the group.

These three algorithms are presented in this chapter along with simulated results, analyses and properties.

3.1 The First Method: Messages Counting (MC)

3.1.1 Algorithm

The method proposed by Melhuish et al. has interesting features. It is a distributed and simple way to estimate the group's size, based only on a very simple assumption:

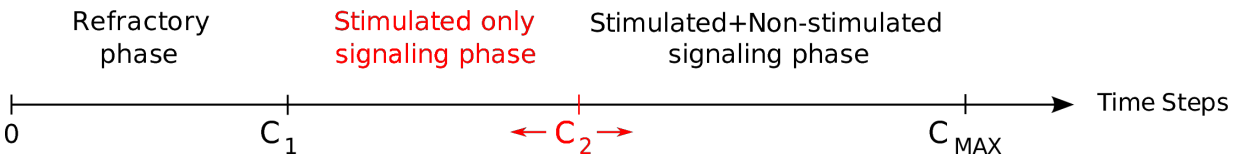


Figure 3.1: The proposed three-phase cycle. The position of C_2 varies according to the time elapsed since the last emitted signal and is unique for each robot.

the robots should be able to perceive the signals of their neighbors. Unfortunately, as seen in Section 2.3.3, for real applications the results of the MHH method are too noisy. The error shown by the system, even with optimal parameters, is large and there is no agreement among robots. Furthermore, the estimate displays strong fluctuations. Whereas these fluctuations can be dampened by means of more conservative averages, the system would display less adaptivity and no significant improvements.

As said, this is a consequence of the fact that the algorithm does not give raise to any kind of order into the sequence of emitted signals. This behavior affects the quality of the estimate and, instead, it would be desiderable that those robots that have recently emitted a non-stimulated signal were less likely to emit another one, while robots that did not have the occasion to emit a non-stimulated signal were more likely to emit one. This way, the estimates would be less noisy, because the probability of having exactly $N-1$ stimulated signals between two non-stimulated ones would be higher. Our goal is then to modify the method to impose this order.

Ideally, when the order is perfect (i.e., there is a fixed order in the signaling sequence) the estimate is exactly equal to the group size. This derives from the fact that each robot is counting the signals between two non-stimulated signals, so, if the order is fixed between the non-stimulated signals, there will be exactly $N-1$ signals. Creating this kind of strong signaling order requires a distributed agreement protocol and more powerful communication abilities than those assumed for our robots and it must be recomputed every time there is a variation in group size. Nevertheless, with a simple modification to the original method, we can obtain an approximated but satisfactory ordering.

The key idea is the following: between the *refractory* phase and the *stimulated+non-stimulated* signaling phase, we introduce a *stimulated only* signaling phase in which a robot can emit a stimulated signal, but has zero probability of emitting a non-stimulated one. Consequently, as shown in Figure 3.1, a robot is in the *refractory* phase when $c \in$

$[0, C_1]$, in the *stimulated only* signaling phase when $c \in (C_1, C_2]$ and in the *stimulated+non-stimulated* signaling phase when $c \in (C_2, C_{MAX}]$. Furthermore, while C_1 remains constant over time and is set to the same value for all robots, the value of C_2 is allowed to vary. The value of C_2 is maximum and equal to C_{MAX} for a robot that emitted a non-stimulated signal in the previous time step. This means that a robot that has just signaled does not pass through the *stimulated+non-stimulated* signaling phase. After each complete cycle of c, C_2 is gradually decreased subtracting a quantity ϵ until it reaches C_1. Clearly, at the same time, the length of the *stimulated+non-stimulated* signaling phase increases by the same quantity. As a result, every robot has a different length of the *stimulated only* signaling phase and the *stimulated+non-stimulated* signaling phase, proportional to the time elapsed since the last non-stimulated signal. The decrease of C_2 continues until it reaches C_1: in this case, the *stimulated only* signaling phase disappears and the phase cycle that results is the same as in the MHH algorithm. The swarm is initialized with $C_2 = C_1$ and $c = 0$. An example of how the length of the two phases varies is presented in Figure 3.2, while the pseudo-code of this algorithm can be found in the Appendix.

To sum up, there could be two different scenarios from the point of view of a single randomly chosen robot. Let us consider a system that has been running for a certain amount of time (i.e. it is not in the initial phase). Suppose that a robot has just received a signal. This robot emits a stimulated signal, sets $c = 0$ and $C_2 = max(C_2 - \epsilon, C_1)$ and increase its stimulated signal count s_t. The considered robot has a ratio between the length of the *stimulated only* signaling phase and the one of the *stimulated+non-stimulated* signaling phase different from the one of any other robot. For a time equal to the length of C_1 there will be no signals, for all the robots are in the *refractory* phase. In the next phase, some robots are in the *stimulated only* signaling phase and these robots are waiting for others to signal. Other robots, instead, are in the *stimulated+non-stimulated* signaling phase, and so they have a probability p of emitting a non-stimulated signal. Let us suppose that our considered robot is the first to emit a non-stimulated signal. It then sets $c = 0$, $C_2 = C_{MAX}$, the group size estimate to $\hat{n}_t = s_t + 1$ and sets $t = t + 1$, $s_t = 0$.

With this mechanism the robots emit with a more precise order, due to the fact that the introduced variable length *stimulated only* signaling phase forces those that have already emitted a non-stimulated signal to remain silent. As shown in Section 3.1.2 this significantly increases the quality of the final estimate.

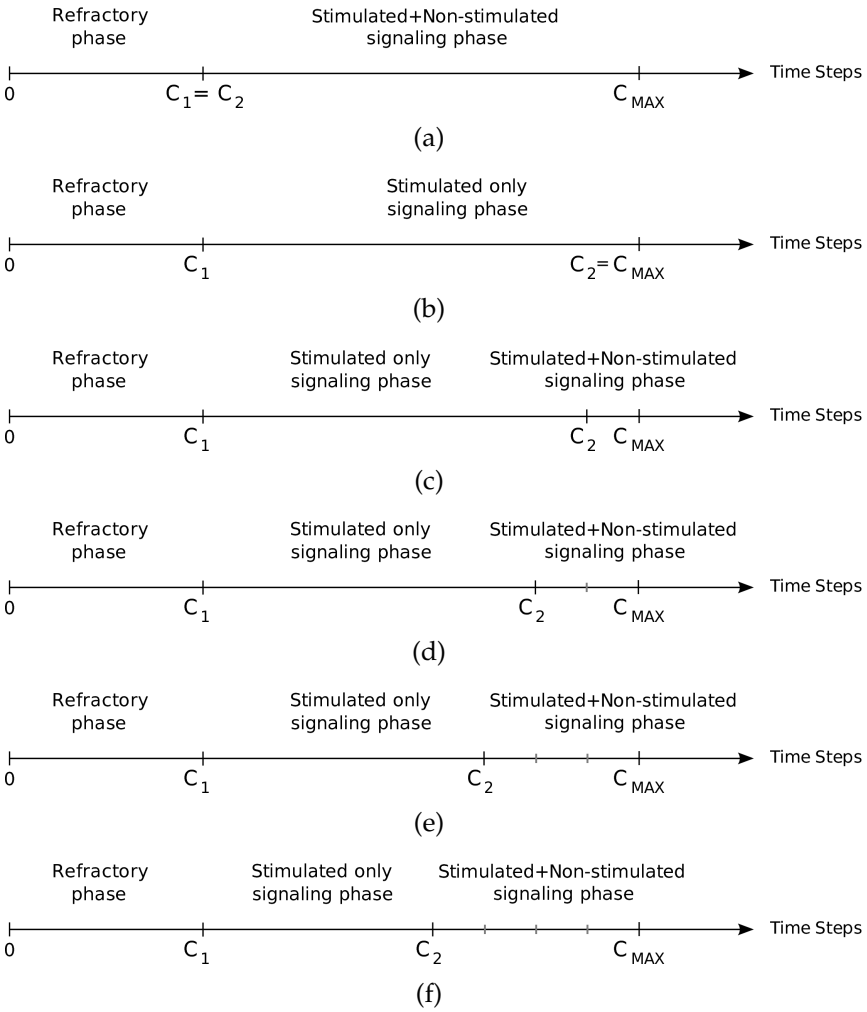


Figure 3.2: The sequence of variation of C_2 for a single robot. The value of C_2 is changed after the emission of a signal.

Parameters

In this mechanism, the choice of ϵ, C_1 and C_{MAX} are critical. Together, they characterize the number of cycles τ needed for the *stimulated only* signaling phase to disappear:

$$\tau = \frac{C_{MAX} - C_1}{\epsilon}$$

With too low a value for τ, the *stimulated only* signaling phase disappears quickly and its benefits vanish – the method behaves similarly to the MHH and suffers the same drawbacks. Hence, a slow decrease is preferable, as it lets room for each robot to signal

in a non-stimulated manner. In general, the larger the swarm size, the larger τ should be to obtain good estimates $\hat{n}_t^i$. To better explain this parameter, let us consider the setup for the experiment we have done. With 25 robots, we set $C_{MAX} = 400$, $C_1 = 20$ and $\epsilon = 10$. This means that $\tau = \frac{400-20}{10} = 38$. An interesting insight about τ is that it can be considered as the number of robots that the method can manage before the advantages of our modification disappear and the method starts to behave in the same way as the MHH method. In fact, a robot that has emitted a non-stimulated signal in the last cycle, and thus has $C_2 = C_{MAX}$, needs exactly 38 stimulated signals to reset C_2 to be equal to C_1. This means that there is space for other 38 robots to signal in the meanwhile. If the robots are less than 38, as in our case, the robots emit a non-stimulated signal even if the *stimulated only* signaling phase has not yet disappeared. The higher the difference between τ and the number of robots present in the system, the better the system works. We can define λ as the difference between τ and the number of robots:

$$\lambda = \tau - N.$$

When $\lambda < 0$, the number of robots is higher than the number of robots that the system is able to manage and thus the overall performance is reduced.

As discussed in Section 1.2, in the typical usage scenario we assume that the initial size of the group is known, so τ can be adjusted to a reasonable value. Furthermore, as casualties due to failures reduce the group's size, λ increases, thus improving the quality of the estimates.

The downside of a high τ is that the total time of the cycle, and thus the time necessary to obtain the estimate, increases. As a consequence, the time needed by the system to obtain an usable and stable estimate is longer. While this is comprehensible, because we need more time to count a larger group of robots, this is a limit for the scalability and usability of the system. As a further improvement of the system, we could implement an adaptive method that autonomously increases τ when λ is close to zero. In this way, the system could tune its parameters autonomously to ensure better performance.

Moreover, the insertion of the *stimulated only* signaling phase renders the choice of p noncritical. As discussed in more detail in Section 3.1.3, although an optimal value for p exists, even suboptimal values do not result in a significant degradation of performance.

With the objective to find the best set of parameters an optimization process has been done on the system using grid search. Figure 3.3 shows the mean relative error for a group of 25 robots. As it is possible to see, the error rate is lower than the error

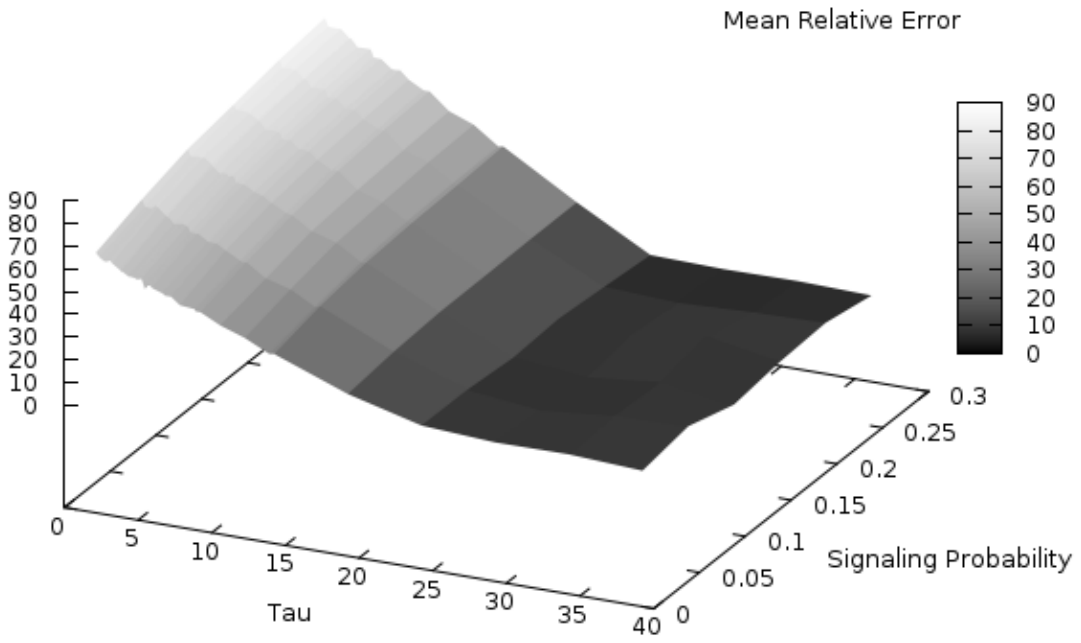


Figure 3.3: The mean relative error as parameters are changed. The error decreases greatly as τ increases and in a less significant way as p decreases.

obtained with the MHH method. The error decreases as C_{MAX} increases and as the signaling probability p increases.

3.1.2 Results

Experiments show that the proposed modification alone significantly improves the quality of the estimation. The mean relative error E obtained with the best parameter values is 1.2%. A run of the algorithm is shown in Figure 3.4. The graphs show that the MC method lowers the overall error in a significant way. Furthermore, the estimates are stable and agreement across the swarm is very high. The behavior of the estimation process presents an initial phase in which the estimate is rising. This is due to the fact that in the beginning not all the robots have flashed yet and so the overall estimate is under the correct one. After two non-stimulated signals the estimation error is already under 20%.

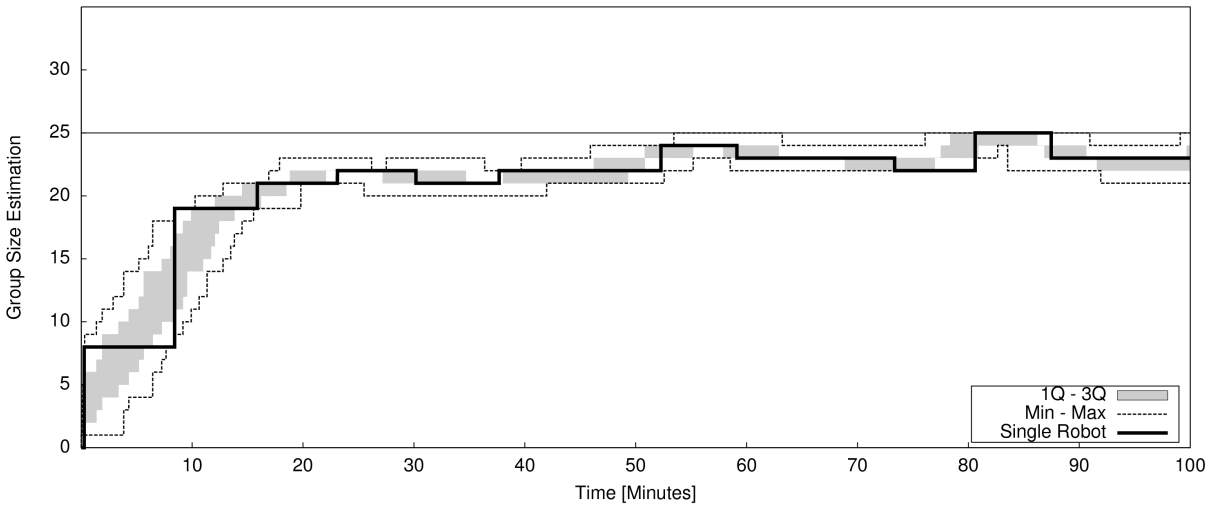


Figure 3.4: Results of the message counting algorithm. The thin dotted line delimits the maximum and minimum estimates in the swarm; the gray area corresponds to the estimations falling into the first and third quartiles; the black solid line is an example of estimate of a single robot.

If we consider the series $S(k)$ of the IDs of the robots that emit a non-stimulated signal, as we have done in section 2.3.3 for the MHH method, we see that our modification imposes on $S(k)$ a periodicity that was absent in the original method. Table 3.1 shows the IDs (from 0 to 24) of the robots when they emit a non-stimulated signal. Consistently with the analysis done for the MHH method, the output of the algorithm is here divided in columns of length 25 to better show the presence of order.

A better way to compare the two methods regarding order in the signals is through spectral analysis. The spectrum in Figure 3.5(b) shows a high peak for frequency $1/N = 1/25 = 0.04$ and subsequent smaller peaks for frequencies $k/N, k \in \mathbb{N}, k > 1$. Each peak corresponds to an increasing part of the series $S(k)$ overlapping with itself. The spectrum corresponding to the MHH method (Figure 3.5(a)) does not display that regularity.

As stated, the results obtained are better than those displayed by the MHH method. However, in the MC method, individual estimates still display slight fluctuations around the equilibrium value. Fluctuations are undesirable because they prevent the robot from ultimately deciding about the swarm's size. For a robot to take a decision based on the swarm's size, the estimation needs to stay constant for a sufficient number of successive observations, beyond which the robot considers the estimate definitive. As discussed in Section 3.1.3, averaging can help reduce fluctuations, thus making the estimates more stable.

	Cycles				
IDs of the robots emitting a non-stimulated signal	0	0	0	0	0
	24	24	24	24	24
	5	20	20	20	20
	20	5	5	21	5
	22	22	21	5	21
	21	21	22	22	22
	18	18	18	18	3
	3	3	3	3	18
	2	2	2	2	2
	9	9	6	6	14
	6	6	14	14	6
	4	14	9	4	4
	16	4	4	9	9
	14	16	16	16	7
	8	8	7	7	16
	7	7	8	17	17
	17	17	17	8	8
	23	23	23	23	23
	1	1	1	1	1
	15	15	10	10	10
	10	10	19	11	15
	19	19	11	15	11
	11	11	15	19	13
	13	13	13	13	19
	12	12	12	12	12

Table 3.1: IDs order for the MC method. Data is divided into columns of length 25 to better show the periodicity of the signal series.

Table 3.2: Mean relative error of the MC methods with different group sizes.

N	C_1	C_{MAX}	p	ϵ	*Error [%]*	Runtime [s]
25	20	400	0.12	10	1.2	482.7
50	20	1300	0.005	10	5.76	3698.2
75	20	1300	0.005	10	6.35	3973.1
100	20	1300	0.005	10	8.28	4122.4

3.1.3 Properties

Our proposed method displays low level of error and high stability and agreement. However, to use the system in real scenarios, we need to analyze other important properties. We would like the system to be able to work in the correct way also as the number of robots scales and to be able to cope with modification in the size of the group. Moreover, we would like to test its stability to suboptimal parameters. In Section 3.1.3 we propose different averages to obtain better stability on the estimate.

Scalability

In the simulated experiments shown in Figure 2.5, 25 simulated robots are used. We studied the scalability of the system for increasing number of robots. As it is possible to see in Figure 3.6, the method scales well. However, when the size of the group is increased to 75 or 100 and the signaling probability p is kept fixed the estimate worsen. This is caused by collisions in the emission of non-stimulated signals. As the number of robots increases the probability that two different robots emit a non-stimulated signal in the same time step increases as well. This leads to an underestimation because the other robots perceive the two signals as if they were a single one. To solve this problem, it is necessary to lower the probability p to emit a non-stimulated signal. Experiments done with a probability $p = 0.005$ give good results in respect to mean error, stability and agreement. However, the time needed to obtain those results is much longer. In Table 3.2 it is possible to see numerical results for different group sizes. Runtime is computed as the average time needed by a robot to update its estimate, i.e. the time elapsed between two non-stimulated signals of the considered robot.

The results for the experiments with 50, 75 and 100 robots are obtained with $C_1 = 20$, $C_{MAX} = 1300$, $p = 0.005$ and $\epsilon = 10$.

Adaptivity

The modified method proves to be adaptive to changes in the swarm size. Figure 3.7 shows the results of a test experiment where initially 25 robots are used. After the estimates across the swarm successfully stabilize around the right value, a second phase starts in which 5 robots are added (Figure 3.7(a)) or 10 robots are removed (Figure 3.7(b)). These results are obtained using the same setup as in Section 3.1.3. In the first case (group size is increased), the system behaves slightly worse than in the second case (when group size is decreased). This is due to the fact that increasing the group size lowers λ, and vice versa. As explained in Section 3.1.1, a higher τ highlights the benefits of the *stimulated only* signaling phase, thus ensuring better adaptivity. Nevertheless, even without further parameter tuning, the method shows a good adaptivity to changes in group size.

Robustness

We wanted to study the robustness of the MC method to parameter variations. In the same way we found the optimal parameters for both systems in Sections 2.3.3 and 3.1.2 we analyzed the system using grid search with a finer scale around the previously found optimal values. We studied the effect of perturbing the probability to signal p around its best found value 0.12 and the decrease quantity ϵ around its best found value 10. Table 3.3 summarizes the mean relative error obtained in each of these setups. The results show that the error does not change considerably with slight modifications of the parameters, thus giving good performance even in case of suboptimal parameter settings.

Stability

To improve the stability of individual estimates $\hat{n}_i^i$, we have tried several averaging strategies. Table 3.4 reports the mean relative errors obtained with the three different averaging strategies we studied.

As can be seen, the price to pay to improve stability is a slightly increased error and less adaptivity (i.e., the system is slower to react to changes of group size). The choice of whether or not to privilege precision over stability depends on the situation in which the method is employed.

Time Weighted Average The average used by Melhuish et al. is a time-weighted average process based on the past history of the system. At every new estimation, the averaged output is computed adding the most recent values (multiplied by a constant α) and the previous average (multiplied by a constant $1 - \alpha$). The used formula is the same presented in Section 2.3.3:

$$\hat{n}_t^i = \alpha(s_t^i + 1) + (1 - \alpha)\hat{n}_{t-1}^i.$$

Parameter α influences the trade off between adaptivity and stability. The lower the parameter, the more stable the output. The price to pay for stable estimates is that if the group size changes (e.g., due to malfunctioning robots) longer time is needed to stabilize to a new value.

An undesired effect of this averaging strategy is that the robots underestimate the real group size. This is due to the fact that this way of averaging takes into account the complete history of the estimates, and that at the beginning of the process, all robots' estimates are initialized to 1, i.e.,

$$\forall i \quad \hat{n}_0^i = 1.$$

Over time, the estimates progressively grow up to the stable value. Given that the past history is always considered during the process, future estimates are always affected by the initial low values, thus causing the system to underestimate the group size (Figure 3.8). The higher the value of α, the higher the effect of history, so the estimate suffers more and more from the initial estimation value. A proposed solution could be to initialize all the robots not to 1, but to the original size of the group. This simple solution is not always practicable or effective. There could be cases where the initial group's size will be unknown or where we want to measure the size of a sub-group of robots. It must be considered that the problem of reduced adaptability remains as an intrinsic downside of all averages systems.

Moving Average To eliminate the underestimation effect caused by the time-weighted average, a moving average is a possible solution. With this strategy, the next estimate of a robot is calculated averaging its last w observations s_j^i:

$$\hat{n}_t^i = \frac{1}{w} \sum_{j=t-1}^{t-w} (s_j^i + 1).$$

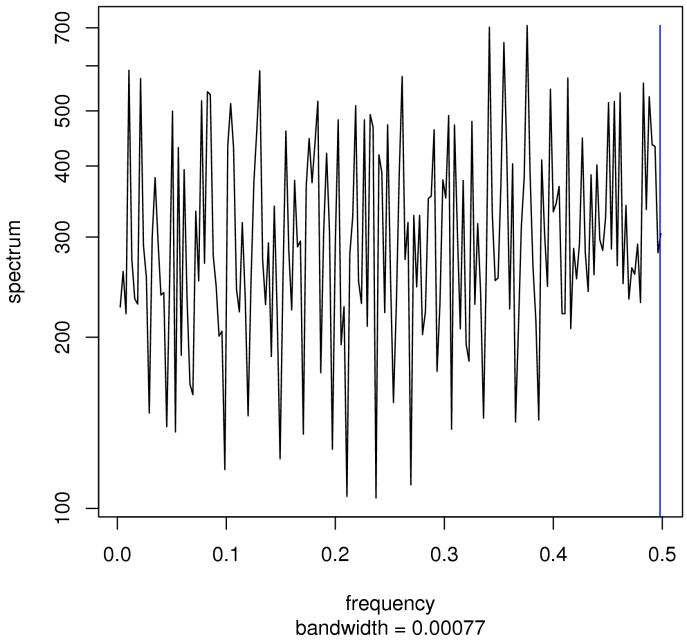
This average solves the underestimation problem, and stability and agreement improve as w increases. On the other hand, increasing w makes stabilization time increase as well, because w estimates are needed to fill the averaging window. Hence, the moving average introduces a trade off between stability and stabilization speed. In our experiments we obtained satisfactory results with the averaging window $w = 9$. Results of various w are shown in Figure 3.9(a).

We also tested a weighted moving average, in which weights are higher for recent observations and lower for older ones. We have computed the average using both linear and exponential weights, noticing no significant improvement.

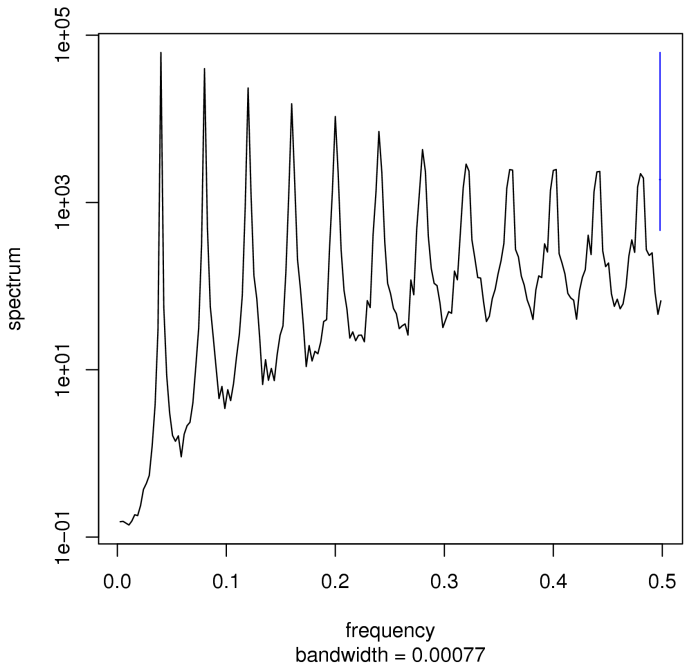
Mixed Average The trade off between stability and stabilization speed introduced by the moving average can be solved with a mixed strategy. Figure 3.9(b) shows that, at the beginning of the process, estimates grow fast towards the stable value. It is only after this phase that averaging is needed. Introducing the moving average at this point allows the system to settle fast around a value and dampen fluctuations from then on, thus retaining the good levels of stability and agreement of the pure moving average.

The benefit of this averaging approach depends on the length of the initial phase when averaging is off. Since robots perform their estimations $\hat{n}_t^i$ only when they signal in a non-stimulated manner, the length of the initial phase can be expressed by the number of non-averaged estimates b before averaging is turned on. The value of b must be high enough to allow the robots to settle around a good estimate. The moving average is a specific case of the mixed average with $b = 0$. In our experiments we found that $b = 2$ is a reasonable value because a single robot needs at least two estimates to achieve a reasonable estimate.

For what concerns the averaging period w, the discussion in Section 3.1.3 still holds. Therefore, $w = 9$ is a reasonable value in this case too. The results obtained with various w and b for this average are shown in Figure 3.9(b).

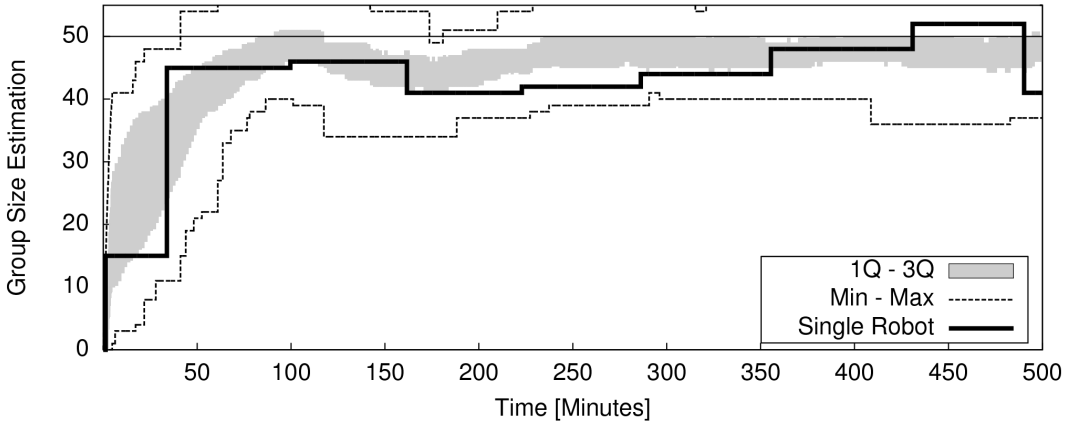


(a) Spectrum of the MHH method.

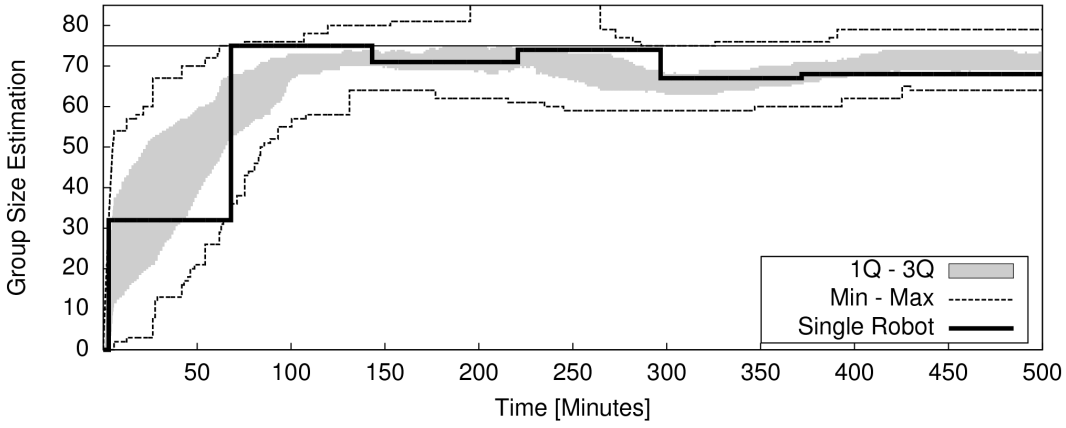


(b) Spectrum of the modified method without averaging.

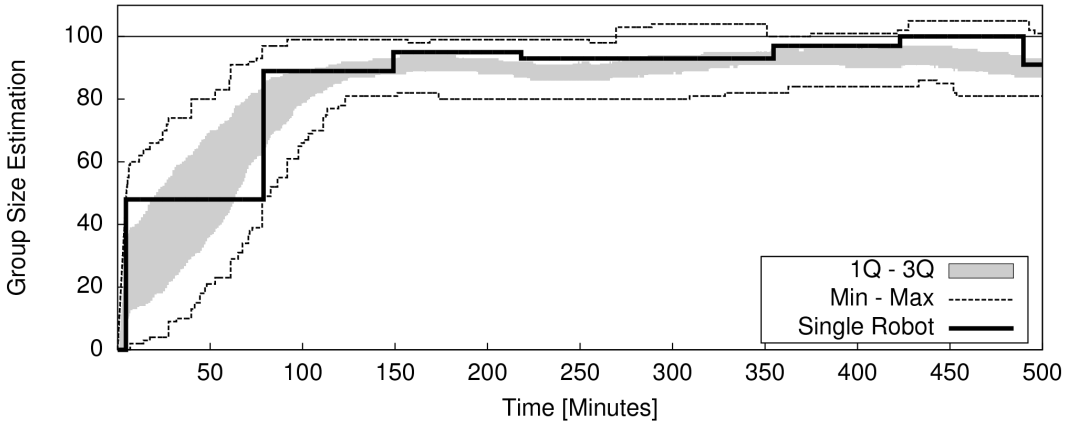
Figure 3.5: Spectral analysis of the sequence of robot ID signaling in a non-stimulated way. Since the robot ID is immaterial, the sequences are analyzed as symbolic time series. The total Fourier spectrum of the symbolic sequence is defined as the sum of the squared modulus of the individual indicator sequence spectra (Afreixo et al. (2004)).



(a) 50 simulated robots.

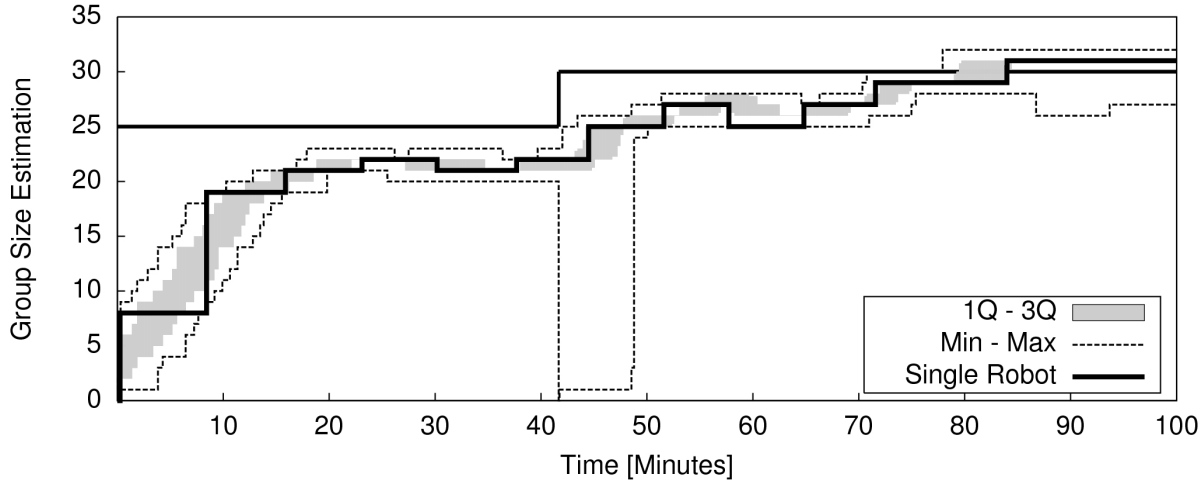


(b) 75 simulated robots.

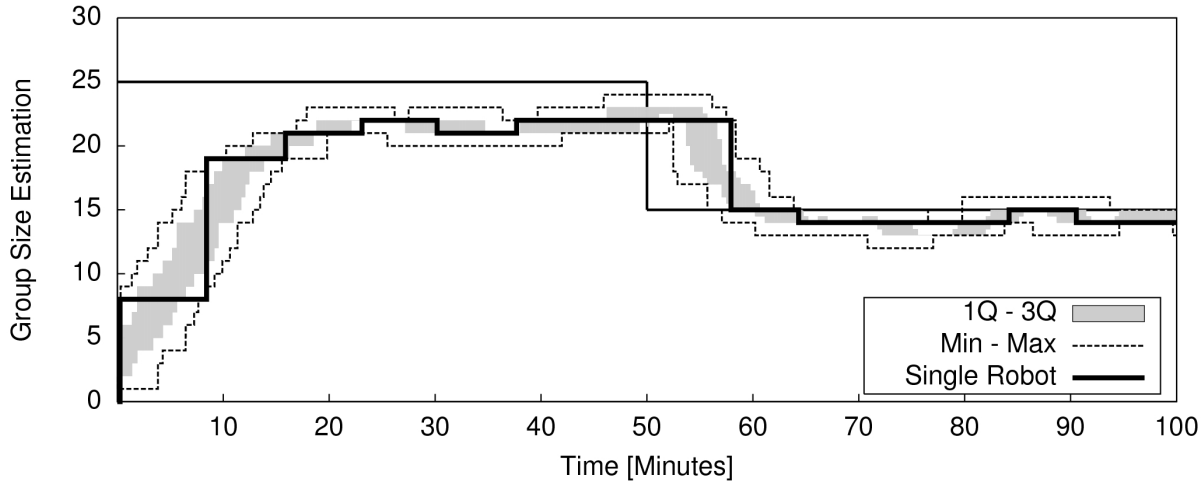


(c) 100 simulated robots.

Figure 3.6: Scalability tests for different group size.



(a)

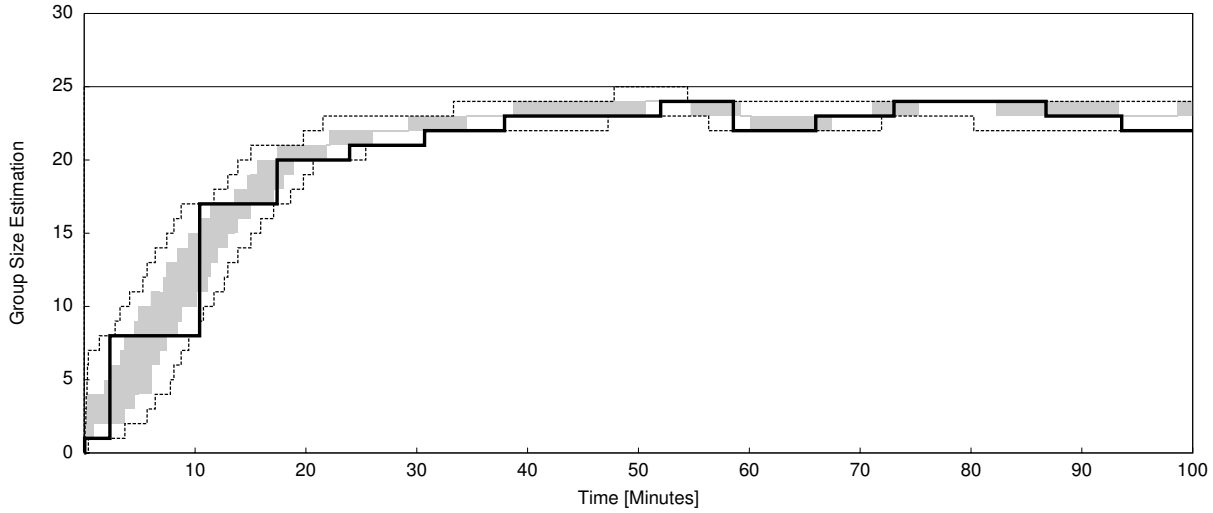


(b)

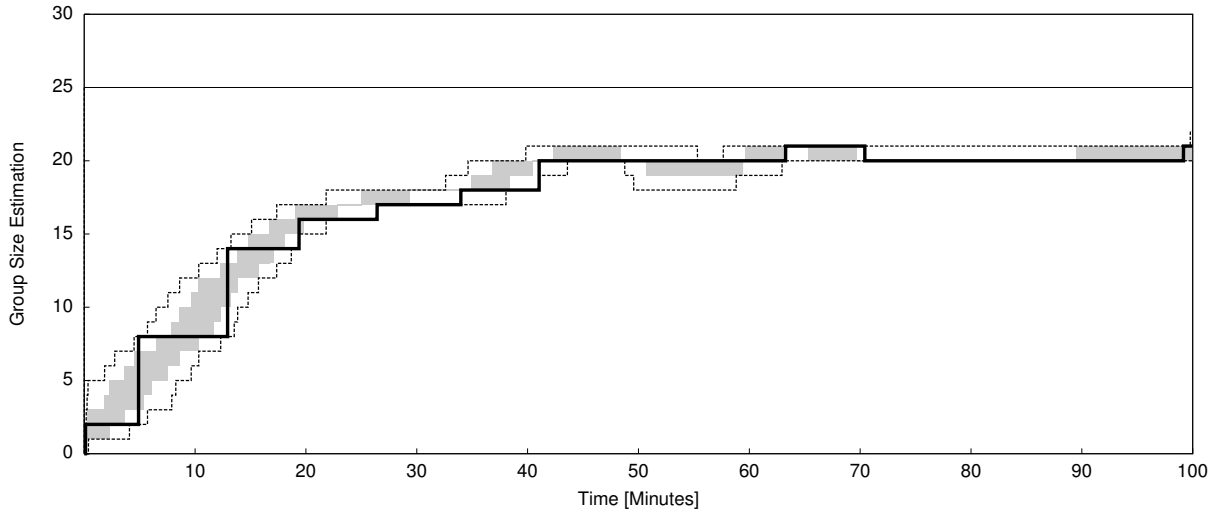
Figure 3.7: Adaptivity tests. In these two-phase experiments, the swarm size is 25 during the first phase. In the second phase, the swarm size is changed to (a) 30 and (b) 15.

Table 3.3: Robustness – mean relative error of the modified method for different values of p, the probability to signal, and of ϵ, the decrease rate of C_2 per cycle.

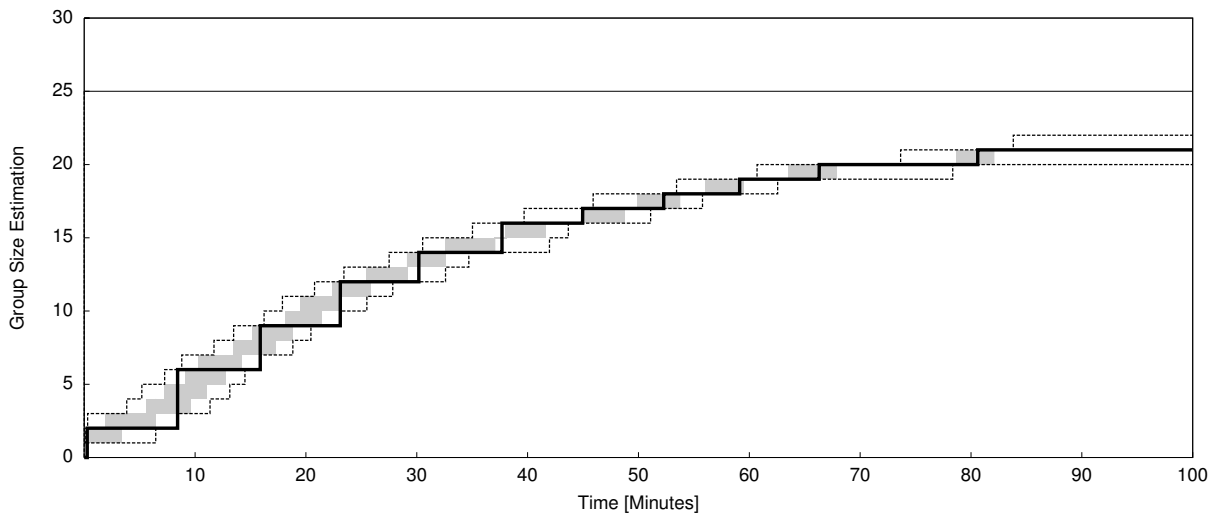
	p									
ϵ	*0.115*	*0.116*	*0.117*	*0.118*	*0.119*	*0.120*	*0.121*	*0.122*	*0.123*	*0.124*	*0.125*
8	2.52	6.04	5.68	3.84	8.8	6.12	8.12	6.28	4.56	9.92	2.96
9	1.92	2.84	4.56	6.2	1.96	6.88	1.96	1.56	2.96	4.28	2.04
10	2.76	3.92	3.52	1.8	1.36	1.2	**0.92**	3.24	1.16	3.16	4.64
11	2.48	4.72	3.76	3.96	1.56	3.12	3.44	3.12	2.28	1.08	3.2
12	2.24	2.76	3.56	3.76	1.12	3.52	4.16	4.28	3.96	4.6	3.8



(a) $\alpha = 0.65$

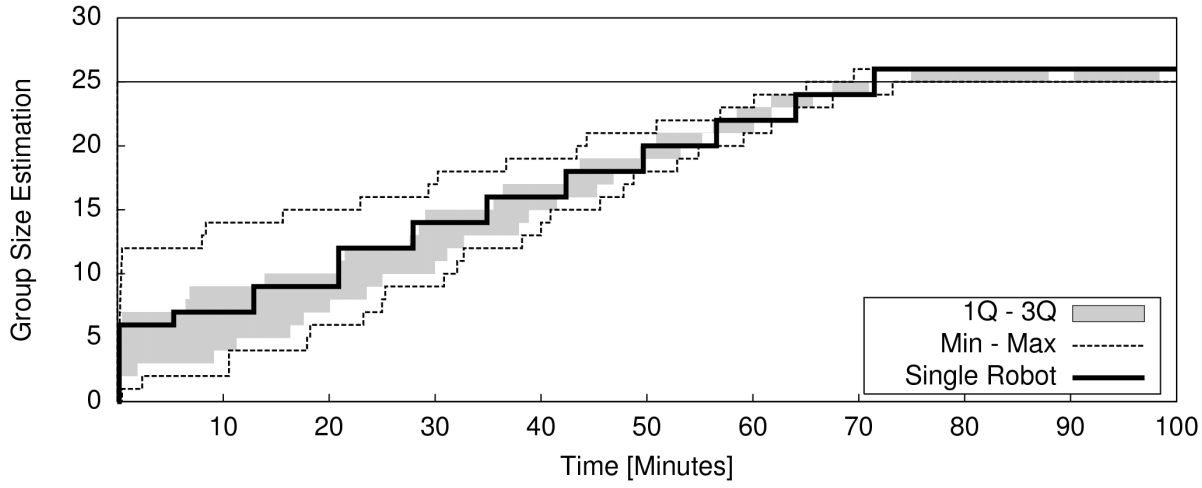


(b) $\alpha = 0.45$

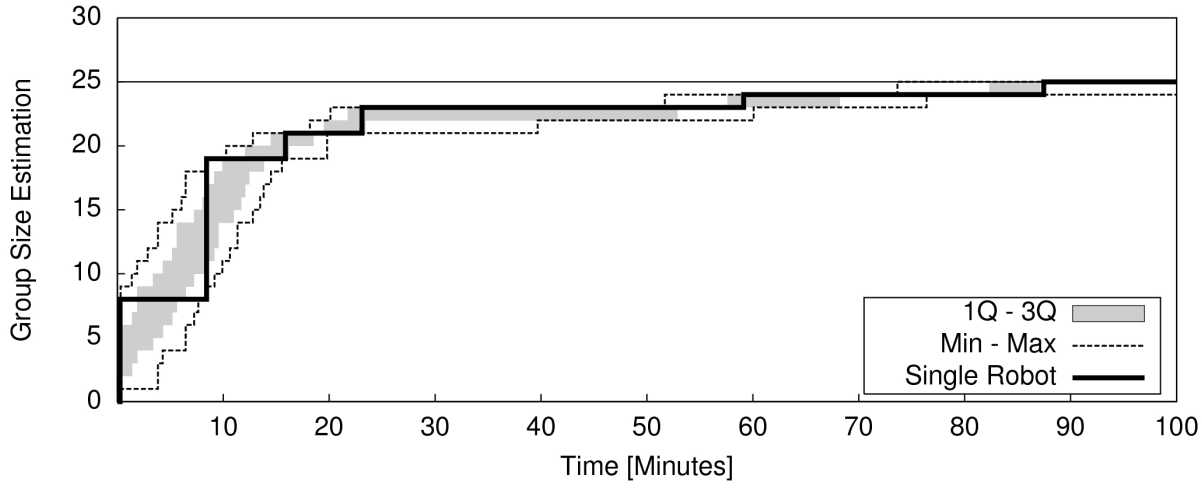


(c) $\alpha = 0.25$

Figure 3.8: Time weighted averages with various α values.



(a) Moving average ($w = 9$).



(b) Mixed average ($b = 2, w = 9$).

Figure 3.9: Comparison of the moving average and the mixed average. It is possible to see how the mixed average solves the speed problem of the moving average.

Table 3.4: Mean relative error of the discussed methods when parameters are set to optimal values.

Method [Averaging Method]	C_1	C_{MAX}	p	ϵ	α	b	w	*Error [%]*
MHH [Time-Weighted Average]	20	260	0.01	-	0.85	-	-	58.24
MC [No Average]	20	400	0.12	10	-	-	-	1.2
MC [Time-Weighted Average]	20	400	0.12	10	0.85	-	-	9.60
MC [Moving Average]	20	400	0.12	10	-	-	9	3.68
MC [Mixed Average]	20	400	0.12	10	-	2	9	4.29

3.2 The Second Method: Time Measuring (TM)

3.2.1 Algorithm

The MHH method is an algorithm for group size estimation that counts the signals received by the robot over a suitable amount of time. With the modification proposed in Section 3.1, we have been able to greatly improve the quality of the estimate but the runtime of the algorithm is generally quite high. A study of the parameters showed a trade-off between accuracy and runtime: by modifying the parameters, in particular the signaling probability p, we manage to decrease the runtime but the resulting relative mean error is higher. This is intuitively an intrinsic aspect of group size estimation and measuring process in general: in order to obtain better results we need more time for the measuring process and vice-versa.

There are cases in which a reduced runtime is more important then accuracy: when dealing with large groups of robots (orders of hundreds) using an algorithm like the one presented in Chapter 3.1 is not feasible in realistic scenarios. A robot like the one considered in our simulation has a battery life of about 90 minutes and thus every method for group size estimation, to be usable, must have a runtime inferior to that. Moreover, faster results should be achieved if we consider group size estimation as a basic algorithm for more complex tasks. With the first algorithm this is not possible (Section 3.1.3 shows that the runtime for a group of 100 robots is approximately of two hours).

To reduce runtime we can face the issue from another point of view. As stated in Chapter 2.3.2, the idea behind the MHH method is that, if every robot emits a signal with probability p, the frequency of the signals is related to the size of the group. One way to estimate the frequency of the signals is to count the signals received over a suitable amount of time, as done in the previous method. A dual approach consists in measuring the time elapsed between two signals, defined as

$$\delta = t_{s_i} - t_{s_{i-1}}$$

where t_{s_i} is the time in which signal s_i was emitted (in seconds) and i indicate a generic signal. Given that each robot emits a signal with the same probability, the longer the time elapsed between two signals, the smaller the group and vice-versa (Figure 3.10).

The algorithm is then the following: each robot has a certain probability p to emit a signal, every time a signal is emitted or received the robot starts an internal stopwatch.

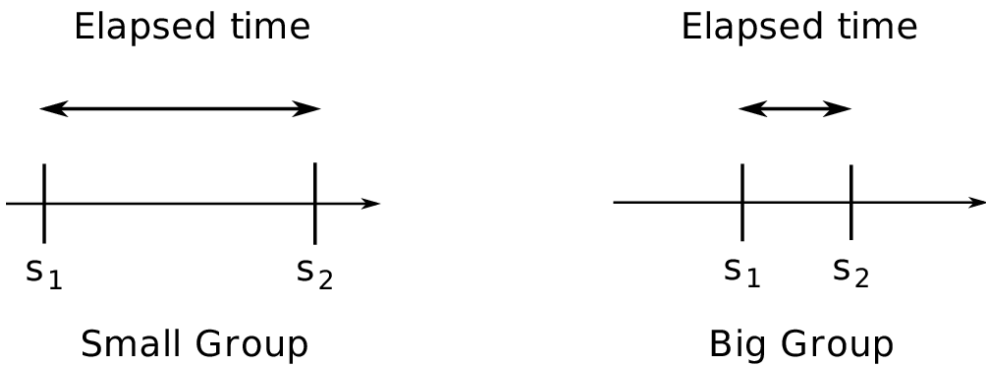


Figure 3.10: The longer the time elapsed between two signals the smaller the group and vice-versa

The robot thus measures the time elapsed between a signal (emitted or received) and another. A number W of measurements are taken and averaged in order to dump errors and fluctuations to improve the estimate. To derive the size of the group from the the elapsed time we need to analyze the probability characteristics of the system.

One robot emits a signal with probability p. It is not possible to predict the time distance between two signals, but it exists a relation between the signaling probability and the period of the signals. We define the pseudo-period as $\frac{1}{p}$.

The probability that one robot emits one signal over k time-steps is:

$$f_1(k) = p(1-p)^{k-1}$$

and the associated cumulative distribution function is:

$$F_1(k) = 1 - (1-p)^k.$$

In a group of N robots, the probability to observe a signal in a time-step, i.e. the probability that at least a robot emits a signal in a given time-step, is:

$$q = 1 - (1-p)^N.$$

This means that the probability that one robot in the group emits a signal in k time steps is:

$$f_n(k) = q(1-q)^{k-1} = (1-p)^{n+k-1}(1-(1-p)^n) = (1+p)^{n+k-1} - (1-p)^{2n+k-1}$$

and thus the cumulative distribution function is:

$$F_n(k) = 1 - (1-q)^k = 1 - (1-p)^{n+k}.$$

Now, to estimate N we first need to estimate q, which can be estimated through the pseudo-period, i.e., the time elapsed between two signals δ. To have a better estimate we consider the average of the last W values of δ:

$$\boldsymbol{\hat{q} = (\frac{1}{W} \sum_{i=1}^{W} \delta_i)^{-1}.}$$

Thus, knowing $\boldsymbol{p}$ and having an estimate of $\boldsymbol{q}$, we can compute an estimate of $\boldsymbol{n}$:

$$\hat{q} = 1 - (1-p)^{\hat{n}}$$

$$(1-p)^{\hat{n}} = 1 - \hat{q}$$

$$\boldsymbol{\hat{n} = \frac{\ln(1-\hat{q})}{\ln(1-p)}.}$$

Therefore, the robots can compute an estimate of the size of the group starting from the signaling probability $\boldsymbol{p}$ and the measured time intervals. The computed estimate suffers from fluctuations caused by imprecise time measurements and by the probabilistic nature of the system but the estimate is obtained is a very short time, as short as the delay between to messages. The pseudo-code of this algorithm can be found in the Appendix.

A common problem of this method and the signal counting method is that we need to carefully tune $\boldsymbol{p}$ in order to avoid the situation in which two robots emit a signal in the same time step or in time steps that are very close. In this situation, the two signals are perceived by the other robots as one, so the estimate of the size of the group will be reduced. To avoid signal overlapping we set the probability that a signal is emitted after less then $\boldsymbol{\beta}$ time steps to be equal to $\boldsymbol{\alpha}$:

$$\boldsymbol{F_N(\beta) = \alpha}$$

$$1 - (1-p)^{\beta+N} = \alpha$$

$$(1-p)^{\beta+N} = 1 - \alpha$$

$$(\beta + N)\ln(1-p) = \ln(1-\alpha)$$

$$\ln(1-p) = \frac{\ln(1-\alpha)}{\beta + N}$$

$$1 - p = e^{\frac{\ln(1-\alpha)}{\beta+N}}$$

$$\boldsymbol{p = 1 - e^{\frac{\ln(1-\alpha)}{\beta+N}}.}$$

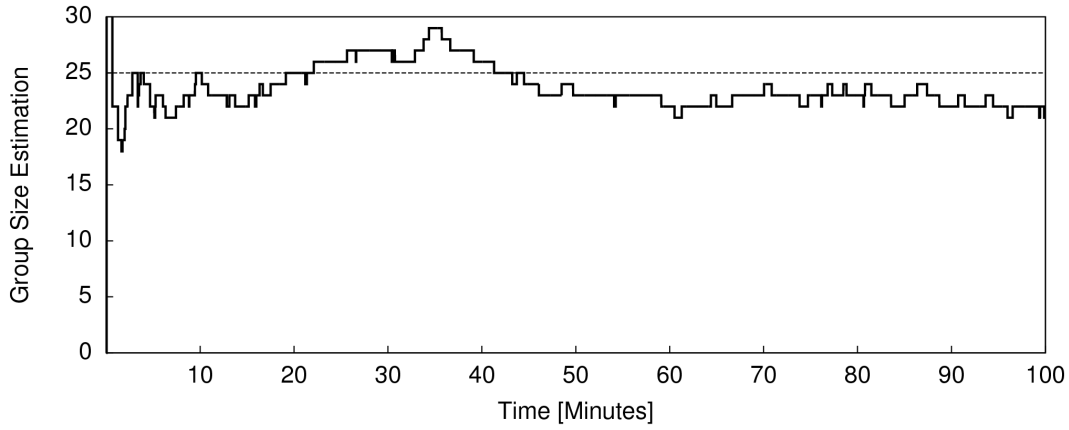
In this way, we can minimize the probability that two signals occur in less then k time steps. Simulated tests show that p has to be quite low in order to minimize the probability of overlapping. Moreover, a longer average interval between signals means that noise is reduced and thus the estimate is improved.

3.2.2 Results

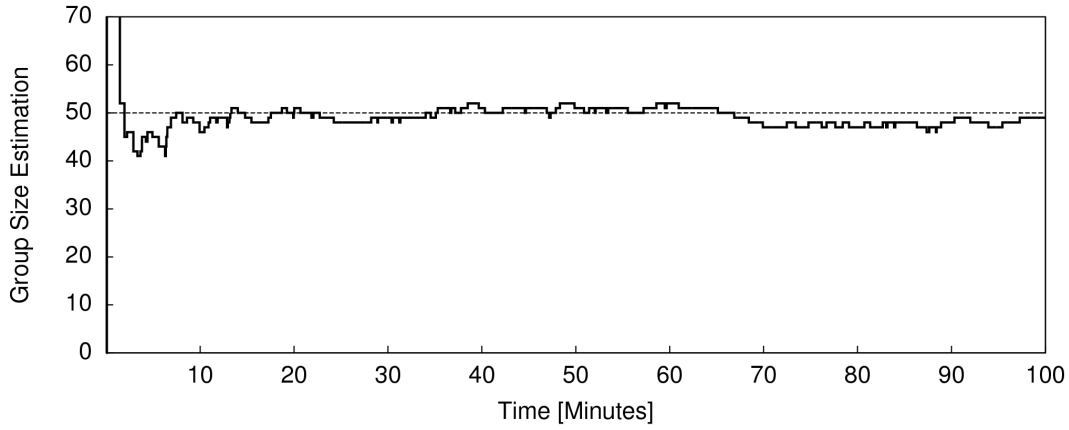
In Figure 3.11 and 3.12 we can see some results of the proposed algorithm. The method here proposed, as said, displays a higher level of error and a significant lower stability if compared to the previous one. Stability can be increased by modifying the averaging window W. As W increases, the average is computed over a bigger number of values and thus fluctuations and noise are reduced giving a more stable estimate. On the other hand, a bigger W means that the estimate need longer time to reach a stable value. This is due to the fact that the robot needs to collect more measurements before the average is close to the correct value. Also, the adaptivity of the algorithm is reduced as more measurement are required to adapt the estimate to the new group size.

Agreement, on the other hand, is very satisfactory. This result is due to the fact that in this method the estimate is not affected by the order of the signals. The only factors that interfere with a total agreement are time delays. In a group of several robots, depending on their disposition in space, the time needed by the signal to be transmitted across the swarm adds error to the measure of the time elapsed between two signals. Hence, the measured time is not δ but, if we define $\hat{\delta}_j$ as the time needed by a message to go from the source to robot j, the real measured time by robot j is $\delta + \hat{\delta}_j$. This means that if a robot is far from the origin of the signal its estimate will be slightly higher then the one of the other robots. Nevertheless, if we suppose the time delay $\hat{\delta}_j$ to be small enough in respect to δ, the estimate will not be severely affected.

As said, the mean error of the method is higher than the one obtained by the previous method but the runtime is greatly reduced. Let us consider the example with 100 robots.

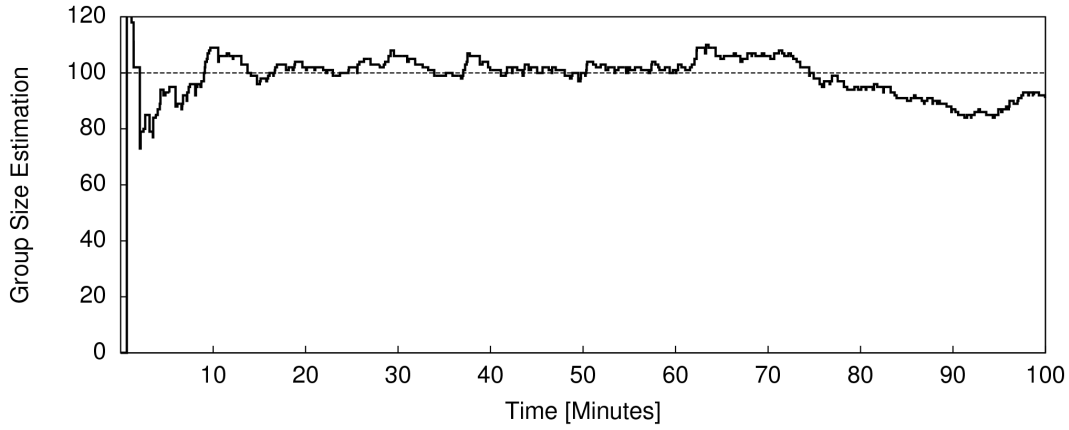


(a) 25 simulated robots.

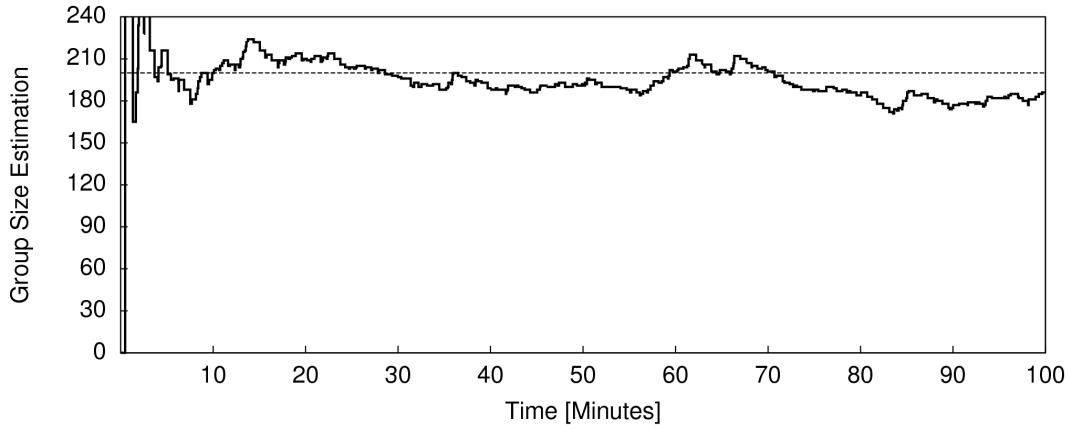


(b) 50 simulated robots.

Figure 3.11: Results for the time measuring algorithm with 25 and 50 robots.



(a) 100 simulated robots.



(b) 200 simulated robots.

Figure 3.12: Results for the time measuring algorithm with 100 and 200 robots.

N	p	w	*Error [%]*	Average δ [s]	$R = W \times \delta$ [min]
25	$3 \cdot 10^{-4}$	100	7.87	14.96	24.93
50	$1 \cdot 10^{-4}$	100	9.65	19.27	32.12
100	$7 \cdot 10^{-5}$	100	3.62	15.32	25.52
200	$3 \cdot 10^{-5}$	100	5.37	17.07	28.45

Table 3.5: Mean relative error and runtime of the time measuring method when parameters are set to optimal values.

With this method, a fairly good estimate, with $W = 100$, is obtained after about four minutes. A similar result with the previous method, while having a lower error level, needs about 130 minutes. Moreover, while in the message counting method it was quite difficult to tune runtime (increasing the signaling probability helps reducing runtime), in this method it is possible to tune it explicitly. A robot needs only δ seconds to compute a rough estimate of the size of the group. However, given that a robot performs an average over W time measures, modification in the size of the group are noticed over longer time. In fact, a robot needs R seconds, where R is defined as $R = W \times \delta$, to re-fill the buffer with new values for a completely new average. To sum up, a rough value of the size of the group will be estimated every δ seconds, while a completely new average will be computed every R seconds.

A numerical comparison between different group size with parameters, error and runtime can be found in Table 3.5.

3.3 The Third Method: School Trip (ST)

3.3.1 Algorithm

In the previous methods, the message counting algorithm and the time measuring algorithm, we have presented two group size estimation techniques with minimal communication requirements. The first algorithm obtains a good accuracy but has a long runtime. The second one has a very short runtime, even with large swarms of robots, but a lower level of accuracy. In order to obtain an estimate that is both accurate and fast to derive, we have developed a third algorithm. This algorithm relaxes the constraint of minimal communication capabilities of the robots and assume that each individual is able to emit and receive a simple message, i.e., an integer. While this can be a limit for some robotic platforms, having more powerful communication capabilities is becoming more and more common in robotics. The robot used in our experiments, a simulated version of the e-puck robot, has the capability to emit and receive a message, through a range and bearing board, in a limited range of around 75 cm. This board allows the robot to exchange a single integer value per message.

The idea behind this third algorithm is inspired by school trips. In a school trip there is often the necessity to keep track of the number of children in the group in order to be sure that no children are lost or left behind. A common technique, normally used every time the pupils are entering the school bus, is to make them count themselves. When a student enters the bus, he says loudly a number, the number of students already on the bus plus himself, the following in the entering queue says the number said by the previous student increased by one and so on. Thus, the first child entering the bus says "one", the second in the queue says "two" (one plus one), the third says "three" (two plus one) and so on. The last student in the queue says its number and that number will be the exact number of children on the bus. At this point the teacher, along with every children, has the information about the size of the group and the process ends. This is a simple, distributed algorithm for estimating the size of the group.

This algorithm is based on two important characteristics. First of all, the children are in a queue and thus they know exactly their turn to say the number. Second, each child pronounces the number only once, when entering the bus.

To adapt the algorithm for a swarm of robots we need to modify the process keeping in mind some differences between the previous example and a robotic scenario. First of all, the robots will not be in a queue and thus there is no explicit order in the signaling

process. As previously discussed in Chapter 3, it is non-trivial to impose a distributed order over a group of robots with only local communication capabilities. Instead of imposing such kind of order, we let the robots emit a signal with a certain probability p in the same way as in the two other algorithms.

Another problem is that the robots do not know when to start the algorithm. In the school bus example we assume that the children are counting themselves every time they are entering the bus. Given that each robot emits a message only once, as in the school bus example, the robots must know when to start the process in order to keep track of variations in the size of the group. A simple way to understand when the process is finished and thus the robot can start a new count is a timeout. Every time a robot receives a message it starts a counter to check if the message is the last one, i.e. every robot has emitted a message. If the counter exceeds the threshold T_{up} the process is considered to be concluded, the estimate is stored and the process is restarted. The timeout should be long enough to avoid losing messages from the last robots. In our experiments we used $T_{up} = 2000[timesteps]$.

Let us sum up how the algorithm works in detail. In the beginning, each robot has the same probability p to emit a message and we call these robots *active robots*. Once a robot emits a signal, it set the probability to 0 and we call these robots *passive robots*. The emitted message is the current count plus one, i.e., the number of robots that have already emitted a message plus the robot that is currently emitting a message. When a robot receives a signal, it updates the counter with the received message and retransmits the same message to propagate it. Eventually all the robots have emitted a message, becoming passive robots, and thus the algorithm finishes. A graphical representation of the process can be seen in Figure 3.13. The pseudo-code of this algorithm can be found in the Appendix.

In this method the problem of overlapping messages is even more problematic than in the other two. When two or more messages are overlapping, the robots perceive them as a single one and thus the estimate is reduced by the number of overlapping signals. In this method, the estimate is not influenced by noise or probabilistic fluctuations and thus if there are no overlapping signals the estimate will be exactly the real size of the group. In order to avoid overlapping signals, we need to set p to a quite low value, especially with large groups (experiments showed that a good value is $p \leq 1 \cdot 10^{-4}$). In this way, we can minimize the probability that two robots emit a message in the same time step. However, a downside of setting p to a very low value is that runtime is

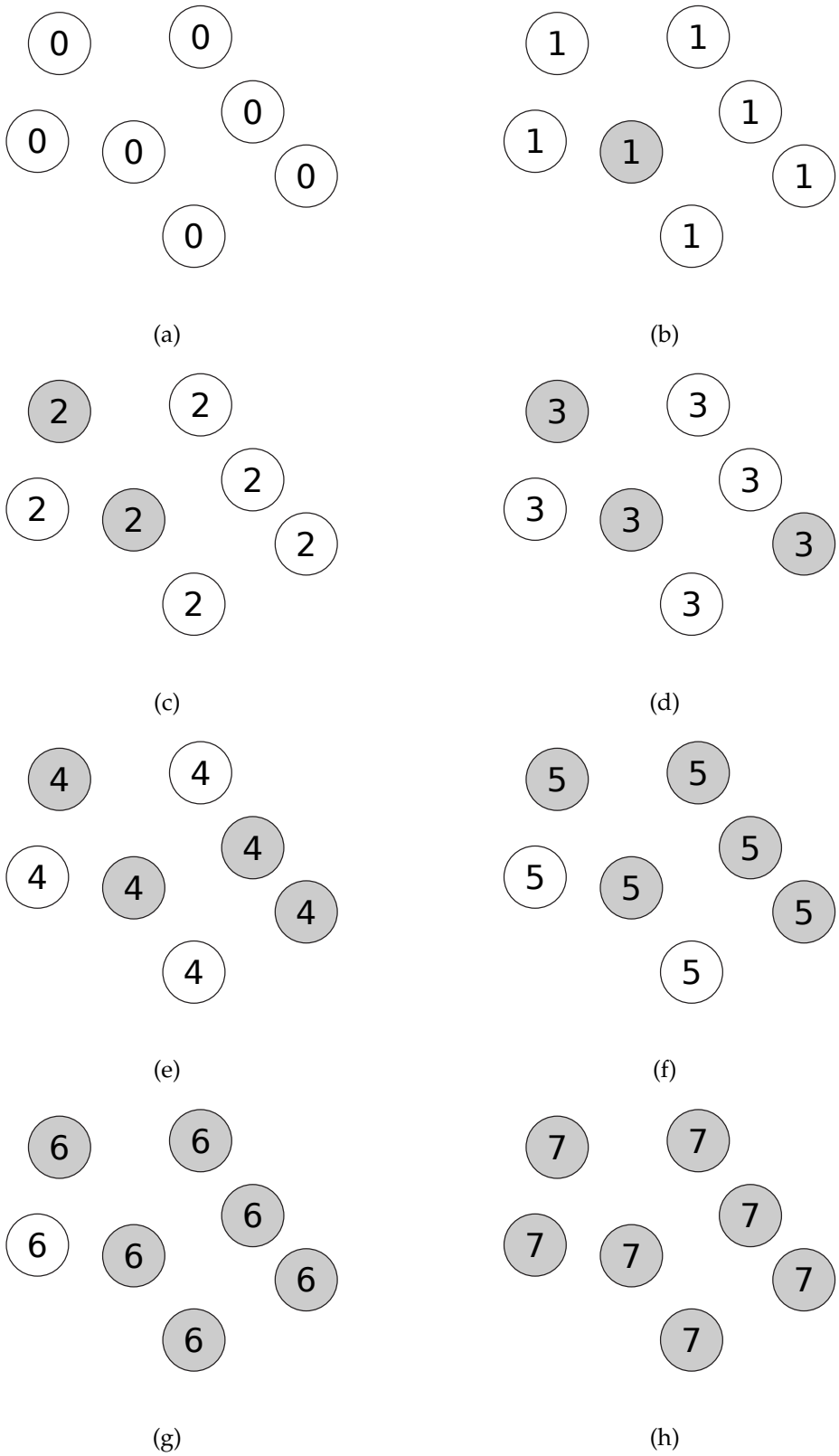


Figure 3.13: A graphical representation of the School Trip algorithm. Robots emit a message containing the actual estimate and then set their probability to emit a message to 0. In the figure, white circles are active robots, while gray circles are passive robots.

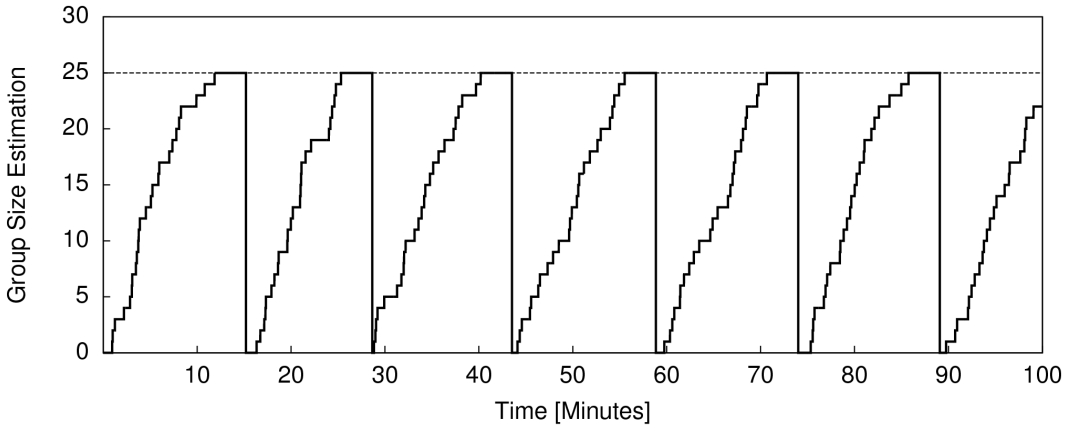
increased.

While in the beginning of the estimation process it is desirable to have a low p value, to avoid overlapping of messages, as more robots emit a message, the number of active robots decreases. This means that while in the beginning we observe a signal with a high frequency, given that there are more active robots, as the algorithm proceeds there will be more passive robots, thus less signals and longer silent periods. We implement a strategy to reduce runtime by minimizing silent periods: the robots dynamically increase the signal probability p as the time between two messages increases. In the beginning, every robot has the same signaling probability p. After the reception of a message each robot starts a counter. This counter has two thresholds: when the counter exceeds T_{up} the counting process is restarted, i.e., the probability of each robot is set to the original value p, thus all the passive robots become again active. The other threshold is lower than T_{up} and applies only to active robots. When an active robot does not receive a message for T_{act} time steps, it increases its probability p. In this way, as the number of active robot decreases, the probability that these robots will emit a message increases, shortening the silent periods. Again, the increase in probability should be carefully tuned in order to decrease runtime but, at the same time, avoid overlapping messages. Starting with a very low signaling probability, we found that doubling p when the counter exceeds T_{act} gives good results and reduces runtime. To ensure that p does not go up to very high level we have put an upper limit to it.

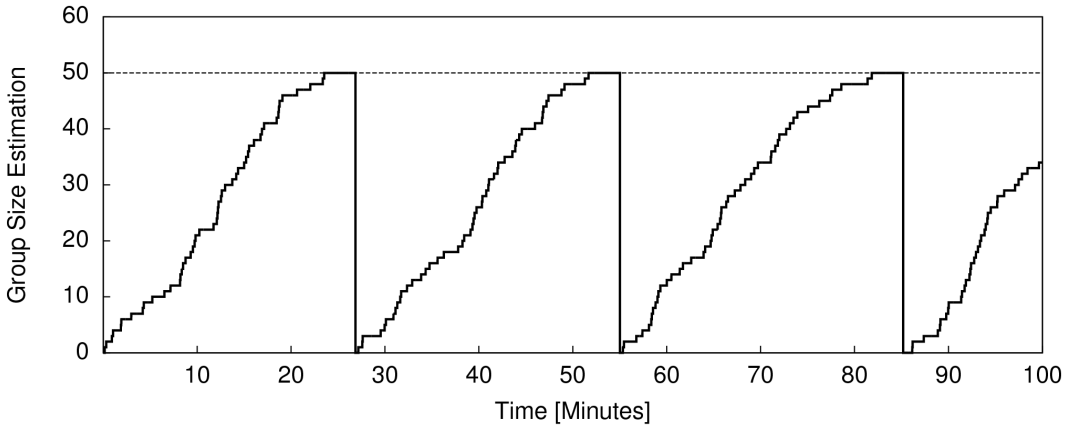
3.3.2 Results

Figure 3.14 and 3.15 shows to see some results of the school trip algorithm with 25, 50, 100 and 200 simulated robots. In the graph it is possible to see how the estimate is built over time until it reaches the maximum amount and the moment in which T_{up} is reached and thus the process restarts. The mean error is generally very low. A good estimate is reached in almost every cycle and independently of the group size. Again, as in the time measuring mechanism, agreement is very high. This is an intrinsic characteristic of the system: as the current estimate is communicated by the signaling robot, as soon as the message reaches every robot, the whole swarm always has the same estimate.

In order to ensure good results in the estimate, we need to impose a very low initial signaling probability p and a quite high ending threshold T_{up}. T_{act} should be set to be high enough to not increase too early the signaling probability. In fact if T_{act} is reached too early, in a moment where many robot are still active, the probability of overlapping

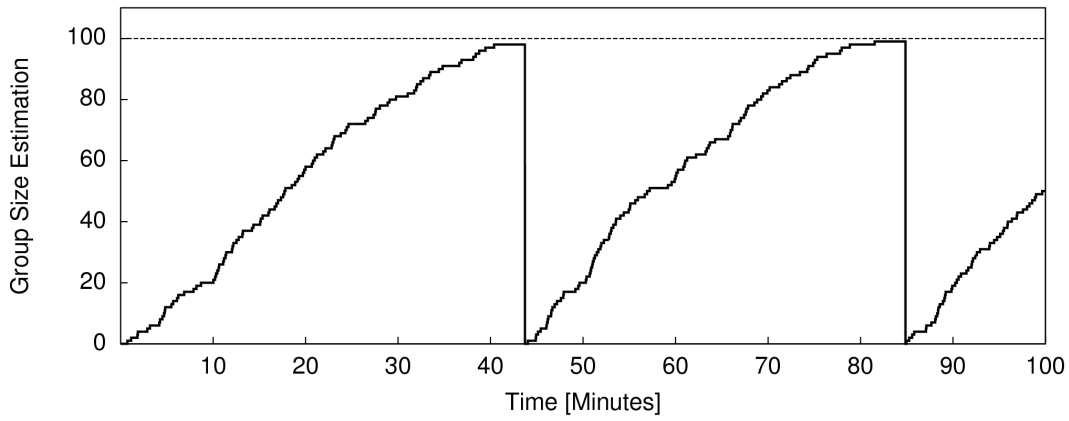


(a) 25 simulated robots.

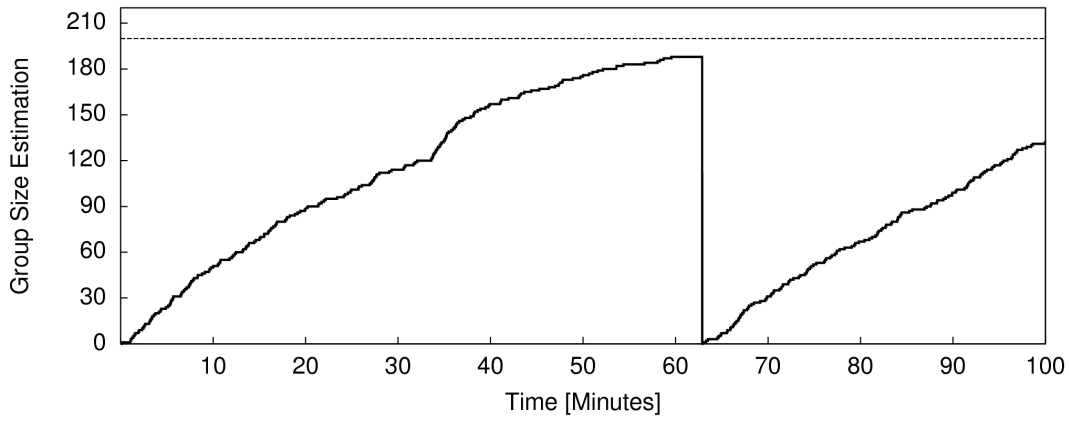


(b) 50 simulated robots.

Figure 3.14: Experiments for the school trip algorithm with 25 and 50 robots.



(a) 100 simulated robots.



(b) 200 simulated robots.

Figure 3.15: Experiments for the school trip algorithm with 100 and 200 robots.

N	p	T_{act}	T_{up}	*Error [%]*	Runtime [min]
25	$1 \cdot 10^{-4}$	500	2000	1.10	14.28
50	$5 \cdot 10^{-5}$	750	2000	1.13	15.38
100	$5 \cdot 10^{-5}$	750	2000	2.60	40
200	$5 \cdot 10^{-5}$	750	2000	4.10	66.6

Table 3.6: Mean relative error and runtime of the school trip method when parameters are set to optimal values to minimize error.

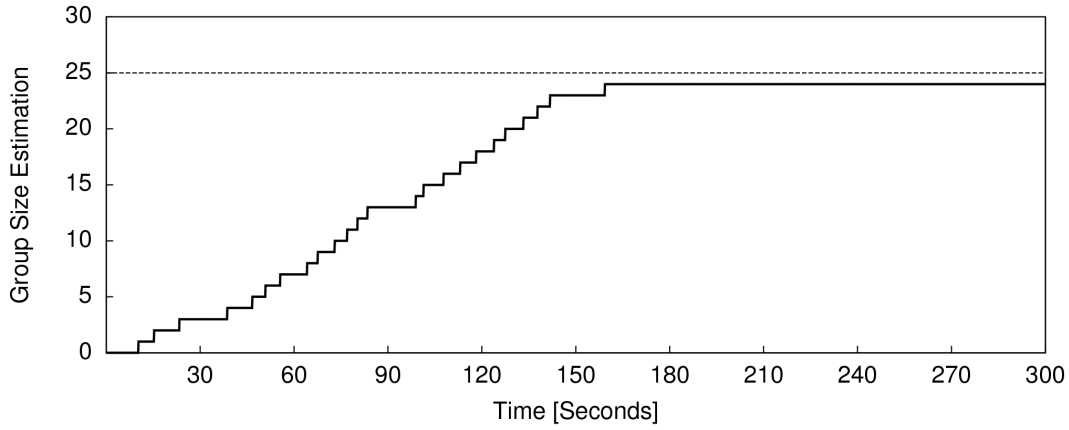
of signals increases. On the other hand, it should be low enough to reduce the runtime of the process. Also, T_{up} must be distant enough from T_{act} to ensure this second threshold to be effective.

The school trip method can be tuned to display a very high accuracy or to obtain quite low runtime. Increasing the signaling probability p and reducing the threshold T_{act} significantly lowers the time needed by the algorithm to come to an end. Figures 3.16 and 3.17 show results when the parameters are set to improve speed. The parameters are shown in Table 3.7. In these graphs, the cycle is not restarted when it comes to an end, i.e. the threshold T_{up} is not used. This is done in order to better show a single estimation cycle in details. With this method we can obtain a fast estimate even with larger group of robots even though we need longer times to estimate the size of a very large group than to estimate a smaller one. Nevertheless, runtime for a group of 100 robots is of about 8 minutes against 130 minutes for the message counting algorithm. The average error is slightly increased but is still reasonable to be used for real applications.

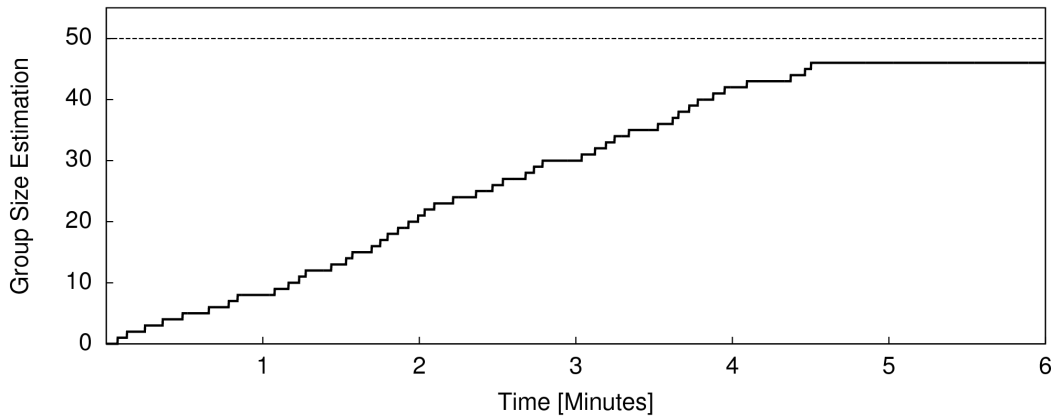
In Tables 3.6 and 3.7 numerical results for the school trip method are presented. The last column is the average runtime of a single cycle. This table shows a set of parameters to obtain a low mean relative error. Table 3.6 shows results when parameters are set to obtain a low error level while Table 3.7 shows parameters to decrease runtime while having still a reasonable accuracy level. T_{up} is not changed in order to present a better comparison based on the other parameters.

3.3.3 Accuracy vs. Runtime

A trade-off between accuracy and runtime is present in all the group size estimation techniques. This can be intuitively considered an intrinsic aspect of most of measuring process. In the first method, the message counting method, parameters can be tuned

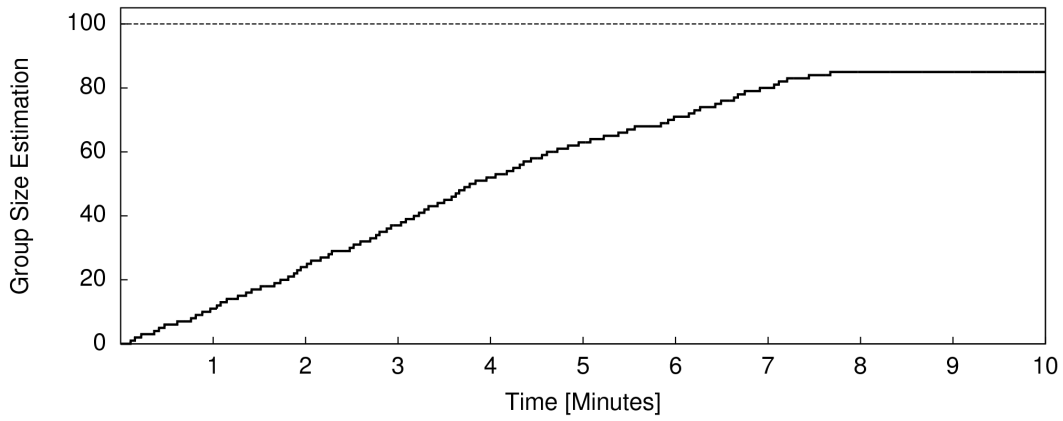


(a) 25 simulated robots.

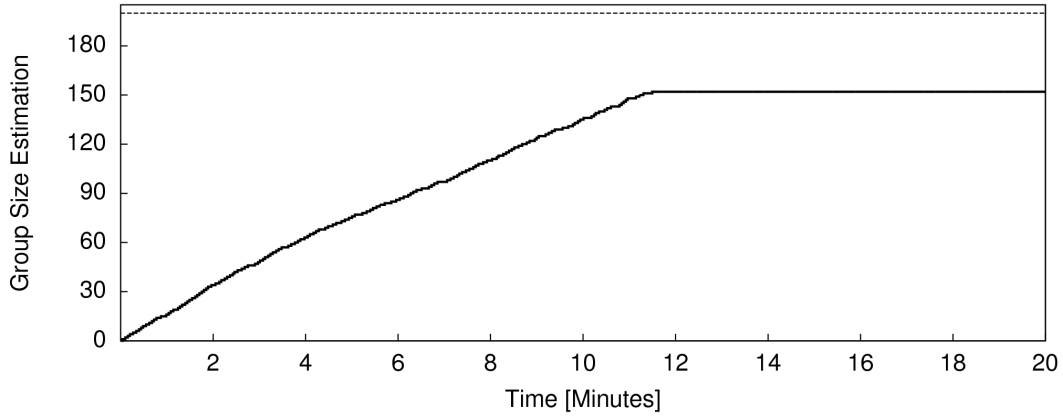


(b) 50 simulated robots.

Figure 3.16: Time elapsed for the school trip algorithm with 25 and 50 robots.



(a) 100 simulated robots.



(b) 200 simulated robots.

Figure 3.17: Time elapsed for the school trip algorithm with 100 and 200 robots.

N	p	T_{act}	T_{up}	*Error [%]*	Runtime [min]
25	$5 \cdot 10^{-4}$	100	2000	6.67	5.88
50	$5 \cdot 10^{-4}$	100	2000	11.57	7.407
100	$5 \cdot 10^{-4}$	100	2000	19.17	10.1
200	$5 \cdot 10^{-4}$	100	2000	29.58	13.51

Table 3.7: Mean relative error and runtime of the school trip method when parameters are set to reduce runtime.

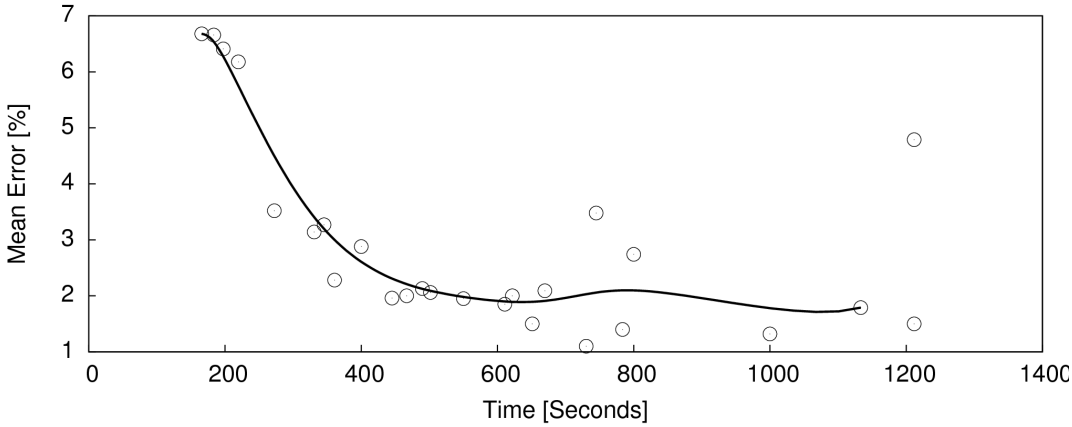
to obtain a low error level in the estimate. On the other hand, it is quite difficult to explicitly tune them to reduce runtime. On the contrary the second method, the time measuring method, is tunable to reach a desired runtime but it is more difficult to obtain a desirable accuracy. This last method, the school trip method, instead, presents characteristics that make it easily tunable both for runtime or accuracy.

As already shown in Tables 3.6 and 3.7 there is a trade-off between accuracy and runtime. This trade-off is better shown in Figures 3.18 and 3.19. In these figures we can appreciate how low runtime bring to higher error levels while low error levels need more time to be achieved. Time is calculated as the average of the number of seconds elapsed between the start of the cycle and the last emitted signal (thus the time taken to reach T_{up} is not considered).

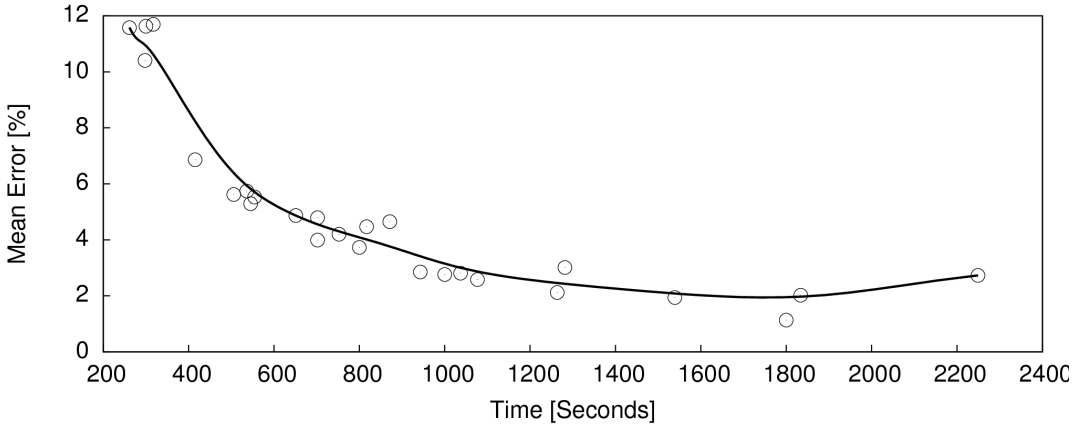
Tuning the signaling probability p and increasing threshold T_{act} result in different accuracy-runtime levels. Higher p and lower T_{act} bring to faster but more inaccurate results. On the contrary, lower p and higher T_{act} result in slower and more accurate results.

3.3.4 Adaptivity

As in Section 3.1.3, we test how the school trip method is able to react to changes in the size of the group. We consider four situations: starting from different group sizes (25, 50, 100, 200) we reduce the size of the group by about the 30% of its size (-10, -20, -30, -60 robots respectively). This could simulate a scenario in which several robots are malfunctioning at the same time because of external causes (e.g. battery depletion, falls into pits or other hazards). Robots are removed after some complete cycles and are removed paying attention to keep the group connected. If the removed robots split the group into two unconnected sub-groups, the messages cannot travel through the whole group and thus the estimate is reduced, i.e. the robots are estimating the size of their sub-group only.

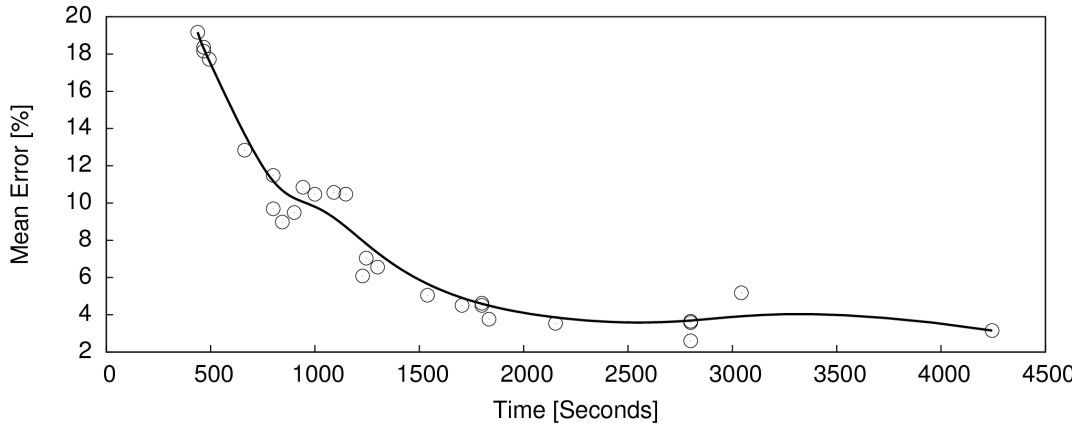


(a) 25 simulated robots.

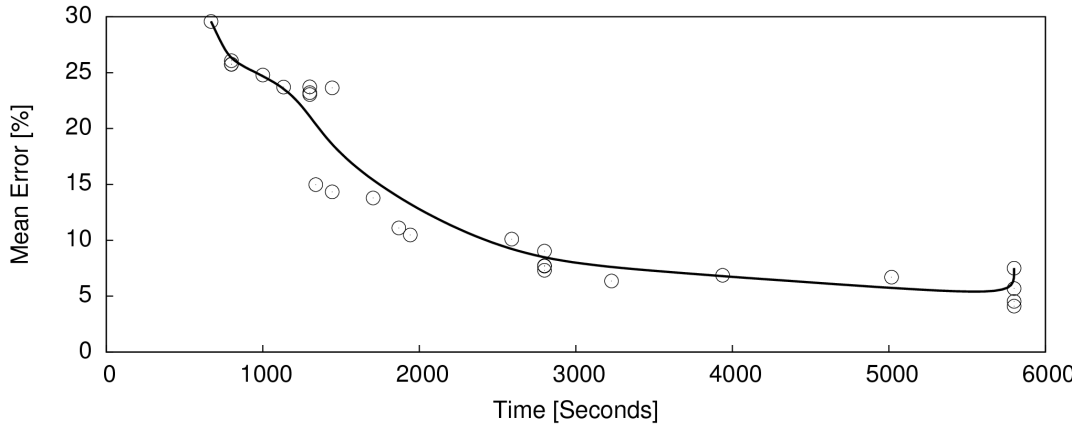


(b) 50 simulated robots.

Figure 3.18: Trade-off between runtime and accuracy with 25 and 50 robots. The points represent experiment results for specific parameter sets. The line represent an interpolation of the data.

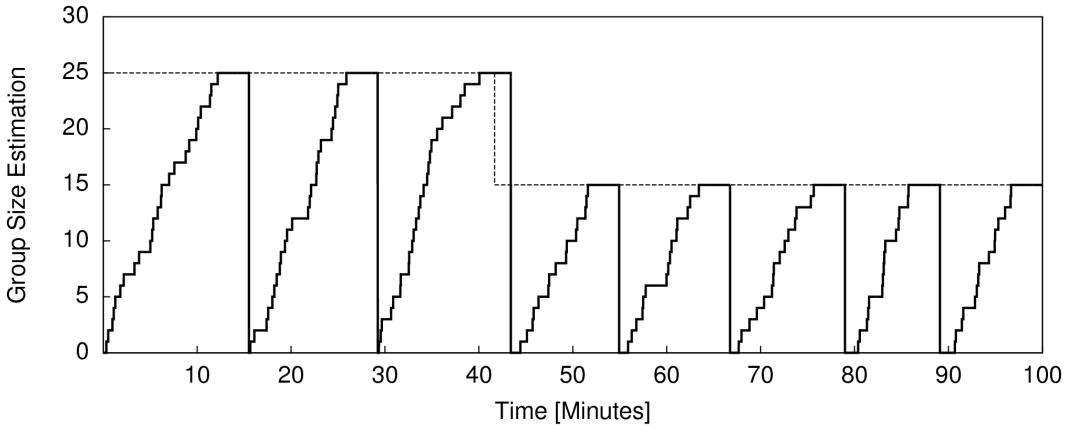


(a) 100 simulated robots.

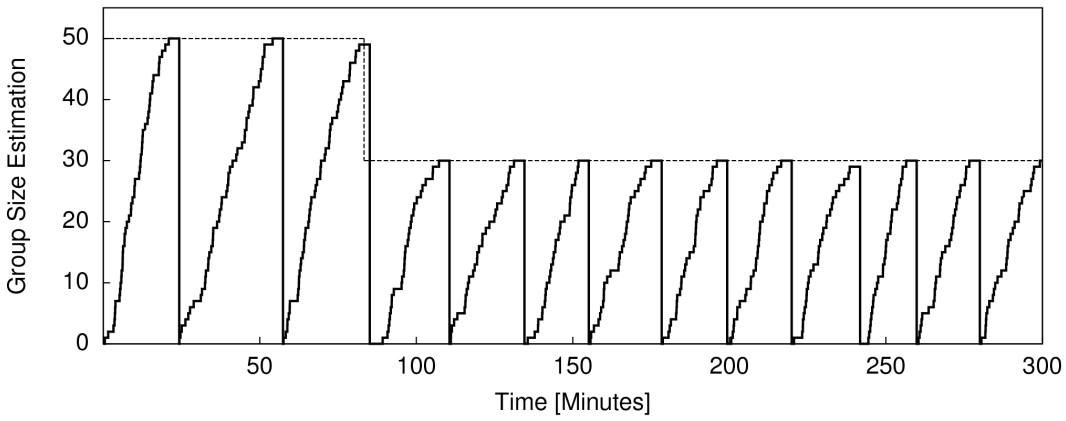


(b) 200 simulated robots.

Figure 3.19: Trade-off between runtime and accuracy with 100 and 200 robots. The points represent experiment results for specific parameter sets. The line represent an interpolation of the data.

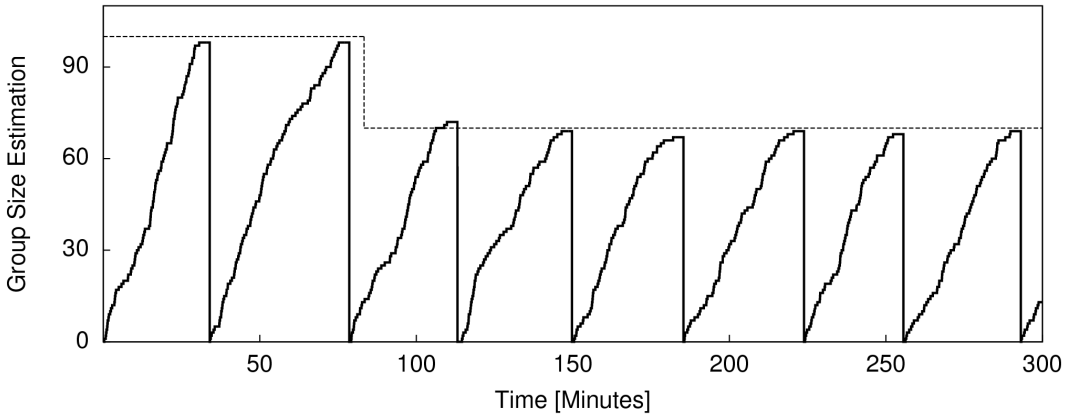


(a) 25 simulated robots. 10 robots are removed after 25000 time steps ($\simeq$ 41 minutes)

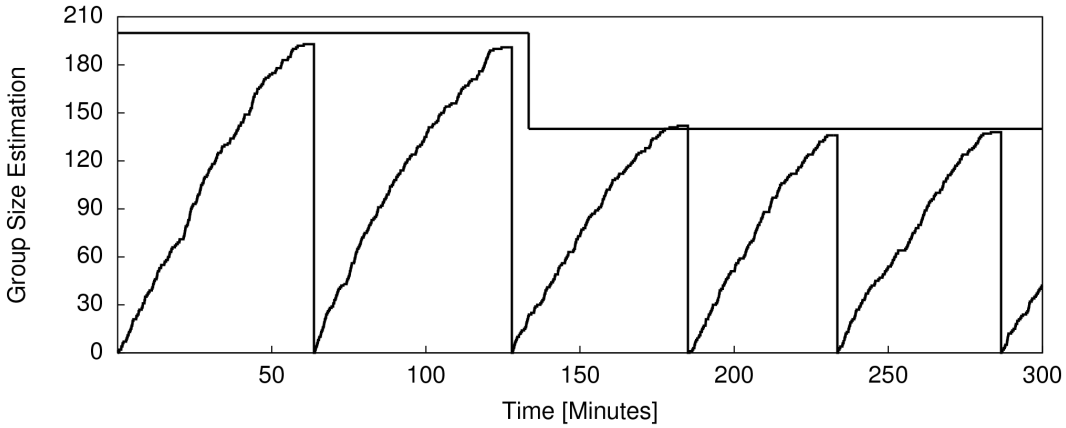


(b) 50 simulated robots. 20 robots are removed after 50000 time steps ($\simeq$ 83 minutes)

Figure 3.20: Adaptivity tests for the school trip algorithm with 25 and 50 robots.



(a) 100 simulated robots. 30 robots are removed after 50000 time steps ($\simeq$ 83 minutes)



(b) 200 simulated robots. 60 robots are removed after 80000 time steps ($\simeq$ 133 minutes)

Figure 3.21: Adaptivity tests for the school trip algorithm with 100 and 200 robots.

As it is possible to see in Figures 3.20 and 3.21, the method is able to adapt correctly to the new size of the group in all the cases. The removed robots are not explicitly chosen between only the active or only the passive ones but are a mix of both groups. This means that if the robots are removed during an estimation cycle we can observe two different situations. If the removed robot is an active one the final estimate will not consider that particular robot as part of the group and thus will be correct. On the other hand, should a robot be a passive one, i.e., it has already emitted a signal, the estimate will consider this robot to be part of the group even if it has been removed. In this case, the estimate will be higher than the real one. This situation happens in the third cycle of Figure 3.21(a). Clearly if the robots are removed between two cycles, i.e. before the following cycle is started, this problem is not present.

3.4 Comparison

The three methods presented in this work share common characteristics. They are methods that are able to estimate the size of a group of robots in a distributed way through only local communication. This means that there is no leader or supervisor and that the robots are only able to communicate with their neighbors. These characteristics make these methods ideal for the use in swarm robotics systems.

In all of the three presented methods, the signals of single robots are emitted according to a certain signaling probability and are propagated with an information wave (Figure 2.6) to ensure that the whole swarm receives the message. These are the common points of the three methods. In the first two methods, the message counting mechanism and the time measuring mechanism, the communication requirements are minimal: a robot only needs to be able to emit and receive a simple signal from its neighbors. This can be done through various means (e.g. visual signals (LEDs + camera)). The last method, the school trip method, needs more complex communication capabilities. The robots need to communicate a single number, thus they need a way to exchange this piece of information. In our experiments we used a simulated version of the range and bearing board for the e-puck robot. With this board a robot is able to emit and receive a message containing a single integer. Other methods, like wireless or Bluetooth communication are possible.

Each method differs from the other in some characteristics and results and there exist no algorithm that is clearly better than the others. The choice of the right approach depends on the application scenario and on the use of the estimate.

We can categorize the methods according to their characteristics and results. The first categorization can be made on the communication capabilities of the robots. If the robots have only very simple communication capabilities or a more complex communication system cannot be used (e.g. it is already used for more important tasks), the school trip mechanism is not usable and thus the choice is restricted to the first two methods.

The second categorization can be made on accuracy versus runtime. All methods present a trade-off between accuracy of the results and time necessary to achieve an estimate. The message counting method is able to give good results in term of accuracy but the downside is longer runtime. The time measuring method, on the other hand, displays reduced runtime while displaying a higher error level and lower stability. The school trip method is tunable with respect to this trade-off, it gives a very high precision

with longer runtime with a certain parameter set or fast results with higher error with a different parameter set. In general, the school trip method outperforms the other two methods in respect to accuracy while being slower than the time measuring method. Of course the price to pay is a more complex communication system. A last observation is that all the experiments have been done in simulation. Tests on real robot are planned for the future and will give more concrete results with respect to the real possibilities of these methods.

In Tables 3.8, 3.9, 3.10, and 3.11 we present a numerical comparison between the three methods. Each table represents the analysis of the method with different group sizes. Error levels are computed in the following way: they are the average of the distance between the real size of the group and the estimated one. In the message counting method, being the only one that present different estimate across the swarm (agreement problem), the error is also averaged over the estimates of single robots.

Measuring runtime is more complex. In the third method, runtime is considered as the time needed for a single estimation cycle to complete, i.e., from the start to when every robot has emitted a signal. In the second method, the runtime is considered as δ, where δ is the interval in seconds between two messages. We must point out that the estimate is averaged over W valued, thus changes in the estimate happen with lower rates. In the table we propose δ because W is a parameter that can be modified according to the specific necessities (trade-off runtime vs. stability). The first method is the more complex to analyze with respect to time. The length of the cycle change from robot to robot according to the distance between two non-stimulated signals. An average of these time distances is thus done across the swarm. 10 time steps correspond to 1 simulated second.

The presented set of parameters are specific for the presented algorithm and are chosen to give the best performance (error or time) while being accurate and general enough to be applicable to different group sizes. The school trip method is presented twice in the tables. The first is with a parameter set chosen to minimize error ("STe" in tables) while the second one reduces runtime ("STr" in tables).

In Table 3.8, we present also results from the original method of Melhuish et al.. In this case runtime is not presented because it varies too much between one cycle and another and between one robot and another. Table 3.11 does not have results from the message counting algorithm because the runtime for this algorithm with 200 robots are too long to be considered meaningful.

N	Parameters	Error [%]	Runtime [s]
MHH	$p = 0.01$, $C_{MAX} = 260$, $\alpha = 0.85$	58.24	-
MC	$p = 0.12$, $C_{MAX} = 400$, $\epsilon = 10$	1.2	482.7
TM	$p = 3 \cdot 10^{-4}$, $W = 100$	7.87	14.96
STe	$p = 1 \cdot 10^{-4}$, $T_{act} = 500$, $T_{up} = 2000$	1.10	857.14
STr	$p = 5 \cdot 10^{-4}$, $T_{act} = 100$, $T_{up} = 2000$	6.67	352.94

Table 3.8: Numerical comparison of the different group size estimation methods with 25 robots.

N	Parameters	Error [%]	Runtime [s]
MC	$p = 5 \cdot 10^{-3}$, $C_{MAX} = 1300$, $\epsilon = 10$	5.76	3698.2
TM	$p = 1 \cdot 10^{-4}$, $W = 100$	9.65	19.27
STe	$p = 5 \cdot 10^{-5}$, $T_{act} = 750$, $T_{up} = 2000$	1.13	923.07
STr	$p = 5 \cdot 10^{-4}$, $T_{act} = 100$, $T_{up} = 2000$	11.57	444.44

Table 3.9: Numerical comparison of the different group size estimation methods with 50 robots.

N	Parameters	Error [%]	Runtime [s]
MC	$p = 5 \cdot 10^{-3}$, $C_{MAX} = 1300$, $\epsilon = 10$	8.28	4122.4
TM	$p = 7 \cdot 10^{-5}$, $W = 100$	3.62	15.32
STe	$p = 5 \cdot 10^{-5}$, $T_{act} = 750$, $T_{up} = 2000$	2.60	2400
STr	$p = 5 \cdot 10^{-4}$, $T_{act} = 100$, $T_{up} = 2000$	19.17	606.06

Table 3.10: Numerical comparison of the different group size estimation methods with 100 robots.

N	Parameters	Error [%]	Runtime [s]
TM	$p = 3 \cdot 10^{-5}$, $W = 100$	5.37	17.07
STe	$p = 1 \cdot 10^{-4}$, $T_{act} = 500$, $T_{up} = 2000$	4.10	4000
STr	$p = 5 \cdot 10^{-4}$, $T_{act} = 100$, $T_{up} = 2000$	29.58	810

Table 3.11: Numerical comparison of the different group size estimation methods with 200 robots.

Chapter 4

Conclusion and Future Work

The ability to estimate the size of a group of robots is, in general, very useful. We know that while the performance of a swarm of robots increases superlinearly with the number of robots composing the swarm, there is a point in which the drawbacks caused by the intrinsic complexity of having large group overcome this benefits. Complexity in control strategies, interference, i.e. robots blocking other robots, and competition for the same resource can lower the performance of the swarm. In general, in many tasks, there is an optimal group size to achieve the best performance.

Moreover, knowing the size of the group can be a valuable piece of information for the robots to choose the best approach to tackle a specific problem. A large group of robots can decide to adopt a faster but riskier strategy knowing that they will still be able to complete the task. A smaller group could adopt a more cautious strategy in order to manage to complete a task. Failure management can also take advantage from group size estimation. In a typical scenario, a known number of robots are deployed for a task and failures are expected as time passes. Through group size estimation, the robots are able to keep track of the number of active robots and deal appropriately with failures: trigger repairing routines, change strategy or demand the intervention of a human operator for backup robots.

In this work we started from the only, to the best of our knowledge, related work on group size estimation. The method developed by Melhuish et al., loosely based on the fireflies synchronization process described mathematically by Mirollo and Strogatz, is a way to estimate the group's size based on signaling. The group's size is estimated by means of counting the number of signals probabilistically emitted by each robot. The idea is that the frequency of the signals observed by the group is related to its size. More

specifically, every robot counts the number of received signals in a time window limited by two emitted signals. Ideally, thanks to the fact that every robot emits a signal with the same probability p, on average, a robot receives $N - 1$ signals where N is the size of the swarm. Unfortunately, the results of this system show a high error rate and no stability and agreement among robots. This is caused by the fact that the system does not impose a global ordering on the non-stimulated signals.

We propose three methods for group size estimation: the first one, called message counting method (MC), is a modification of the original MHH method; the second one, called time measuring method (TM), is the dual method, time between two messages is measured in order to derive an estimate of the size of the group; in the last one, called school trip method (ST), the robot communicate esplicitly the number of messages emitted thus deriving an estimate of the group. We have done an accurate study of the three systems.

For the first method (MC) we started with the Melhuish et al.'s method and we proposed a modification to obtain better results. Our modification introduces a new phase in the cycle in which a robot has probability zero to initiate a signal but propagation of received signals is allowed. Moreover, the length of this phase is not fixed, as in the original method, but its length is proportional to the time elapsed from the last emitted signal: the robot is prevented from emitting a signal if it has emitted a signal in the last cycle. With this simple modification, we are able to obtain an approximate signaling order among robots and thus a better estimate. The obtained quasi-order greatly improves the quality of our estimate lowering the mean relative error and increasing stability and agreement. Moreover, further experiments have shown that the system is able to cope with variations in the group size, i.e. it shows adaptivity and scales quite well. Runtimes for this method are quite high, especially with large swarm of robots. In addition, we have done a study on robustness of the system to parameter changes and proposed several averages to improve stability.

The second method (TM) is the dual method of the first one. In the MC method the number of messages received over a suitable amount of time was used as an estimate of the size of the group. In this method, on the contrary, robots measure the time elapsed between one message and another. The various measurements are then averaged and then an estimate of the size of the group is derived. This method, while displaying a slightly higher error level and a lower stability, has a very satisfactory level of agreement (due to the fact that every robot is measuring almost the same value) and drastically

reduced runtime. Experiments with increasing group sizes (up to 200) showed how this method is able to produce an usable estimate in a very short time.

The last method (ST) relaxes the constraint of minimal communication requirements and allows robots to communicate the number of received messages explicitly. This method, inspired by the method used by teachers to count children in school trips, let every robot emit a message with a certain probability p. Every time a robot emits a non-stimulates signal it increases the common counter by one and then it sets its signaling probability to 0. In this way every robot emits a single message containing the number of messages emitted until that moment. With this mechanism we are able to obtain a good estimate in relative short time even with large groups. Agreement is ensured given that every robot receives the common count. Moreover, by tuning the parameters we can move from a parameter set that gives a good accuracy but in a longer runtime to a parameter set that gives faster result while having a higher error level.

All the three methods are suitable for estimating the size of a group of robots. However, each method has its own characteristics that make it more suitable for specific situations. The ST method provides the best results with respect to error and quite good results with respect to runtime but needs more powerful communication means. The MC method has very good results but it needs quite long time to complete while the TM method has very fast runtime but lower accuracy.

In this work we presented three method for group size estimation in swarm robotics. There is no method that clearly outperfoms the others so the designer of the system has to choose the best technique according to the specific scenario.

4.1 Future Work

We plan to port the system to real robots, specifically the e-puck robot, to test the quality of the work in real scenarios. Further studies are necessary to understand the effect that noise and missed messages may have on the system.

Another interesting work could be to allow the robots to have a self-tuning mechanism of the parameters of the algorithm. A method to automatically tune parameters based on the estimate of the single robot could improve the adaptability of the systems. A possible drawback could be that if each robot has a different set of parameters the system could display low level of agreement or behave in unpredictable ways.

Finally, we want to test these techniques in real scenarios, in which the information

on the group's size is used. We plan to do tests on group size regulation and fault tollerance using the presented methods as a starting point.

Chapter 5

Appendix

5.1 Code: MHH method

In this section we present the pseudo-code for the MHH method. The algorithm was derived from Melhuish et al. (1999); Holland and Melhuish (1997); Holland et al. (1997).

```
/* Initialize the counter to a random number. */
count = getRandomNum();

/* Set the sending flags. */
sending = 0; // Default state.
MYSELF = 1;  // Non-stimulated signal.
OTHERS = 2;  // Stimulated signal.

/* Initialize the counting variables. */
signal_count = 1;              // Count of the received signals.
group_size_estimate = 1;       // Actual estimate.
group_size_estimate_temp = 1;  // Previous estimate.

/* The controller cycle. */
while(TRUE){

   /* Deal a random number for the probabilistic event. */
   random = getRandomNum();
```

```
/* Reset the sending flag. */
sending = 0;

/* Increase the counter. */
count++;

if (count >= C_MAX){
   /* The counter exceed C_MAX.
      Emits a signal
      Do not increase the estimate. */
   emitSignal();}

else{
   /* The counter is below C_MAX. */

   if (count >= C_1) {
      /* The robot is not in the Refractory phase. */

      if (random < SIGNAL_PROB){
         /* The dealt random number satisfies
            the threshold to emit the signal. */

         /* Set the flag to non-stimulated signal. */
         sending = MYSELF;
         /* Emit a signal. */
         emitSignal();
      }
      else
         /* The random event did not occour.*/

         if (checkMsgReceived()){
            /* The robot has received a message. */
```

```
                /*Set the flag to stimulated signal. */
                sending = OTHERS;
                /* Emit a signal. */
                emitSignal();
            }

        }

    /* Reset the count. */
    count = 0;
    }

    if (sending == MYSELF){
        /* Non-stimulated signal. */

        /* Compute the new estimate with the time-weighted average. */
        group_size_estimate =
            (ALPHA * signal_count + (1.0-ALPHA)*group_size_estimate);
        /* Output the group size estimate. */
        print(group_size_estimate);
        /* Save the estimate as a previous step for the average. */
        group_size_estimate_prev = group_size_estimate;
        /* Reset the count of received signals. */
        signal_count = 1;
    }
    if (sending == OTHERS) {
        /* Stimulated signal. */

        /* Increase the count of received signals*/
        signal_count++;
    }

}
```

5.2 Code: the MC method

In this section we present the code for our MC method. The averages are not taken into account.

```
/* Initialize the counter to a random number. */
count = getRandomNum();

/* Set the sending flags. */
sending = 0; // Default state.
MYSELF = 1;  // Non-stimulated signal.
OTHERS = 2;  // Stimulated signal.

/* Initialize the counting variables. */
signal_count = 1;              // Count of the received signals.
group_size_estimate = 1;       // Actual estimate.
group_size_estimate_temp = 1;  // Previous estimate.

/* The controller cycle. */
while(TRUE){

   /* Deal a random number for the probabilistic event. */
   random = getRandomNum();

   /* Reset the sending flag. */
   sending = 0;

   /* Increase the counter. */
   count++;

   if (count >= C_MAX){
      /* The counter exceed C_MAX.
         Emits a signal
         Do not increase the estimate. */
```

```
      emitSignal();}

   else{

      /* The counter is below C_MAX. */

      if (count >= C_1) {
         /* The robot is not in the Refractory phase. */

         if (checkMsgReceived()){
               /* The robot has received a message. */

               /*Set the flag to stimulated signal. */
               sending = OTHERS;
               /* Emit a signal. */
               emitSignal();
         }
         else if (count => C_2)
            /* The robot is in the Listening+Signaling phase */
            if (random < SIGNAL_PROB){
               /* The dealt random number satisfies
                  the threshold to emit the signal. */

               /* Set the flag to non-stimulated signal. */
               sending = MYSELF;
               /* Emit a signal. */
               emitSignal();
            }
      }

   /* Reset the count. */
   count = 0;
   }
```

```
   if (sending == MYSELF){
      /* Non-stimulated signal. */

      /* Store the new estimate. */
      group_size_estimate = signal_count;
      /* Output the group size estimate. */
      print(group_size_estimate);
      /* Reset the count of received signals. */
      signal_count = 1;
      /* Set C_2 to C_MAX */
      C_2 = C_MAX;
   }
   if (sending == OTHERS) {
      /* Stimulated signal. */

      /* Increase the count of received signals*/
      signal_count++;
      /* Decrease C_2 by epsilon or set it to C_1*/
      C_2 = min(C_1 , C_2-EPSILON);
   }

}
```

5.3 Code: the TM method

In this section we present the code for the TM method.

```
while(true){
   if (checkMsgReceived() or (random < SIGNAL_PROB)){
      /* The robot has received a message or has emitted one. */

      time_elapsed = getTimeElapsed();
      emitSignal();

      /* Average last time measure with the other W measures. */
      time_estimate = doTimeAverage(W);

      /* Derive the estimate from time. */
      group_estimate = log( 1 - ( 1/time_estimate)) / (log(1 - SIGNAL_PR

      print(group_estimate);

   }

}
```

5.4 Code: the ST method

In this section we present the code for the ST method.

```
while(true){

   counter++;

   if (checkMsgReceived()){
      /* The robot has received a message. */

      /* Update the count of received signals. */
      tmp_estimate = getReceivedMessage();

      /* Re-emit the message. */
      emitMessage(tmp_estimate);

      /* Start the counter. */
      counter = 0

      /* Check if the counter has reached the threshold. */

   }
   else if (random < SIGNAL_PROB){
      /* The current robot is emitting a signal */
      tmp_estimate++;
      emitMessage(tmp_estimate);

      /* Start the counter. */
      counter = 0
   }

   if ( reached(count,T_ACT)){
         /* Increase probability */
 SIGNAL_PROB = SIGNAL_PROB * 2;
```

```
 counter = 0;
      }

   if ( reached(count, T_UP)){
      /* Restart the counting process */
      SIGNAL_PROB = ORIGINAL_SIGNAL_PROB;
      counter = 0;
      estimate = tmp_estimate;
      print (estimate);
      tmp_estimate = 0;

   }

}
```

Bibliography

Afreixo, V., Ferreira, P., and Santos, D. (2004). Fourier analysis of symbolic data: A brief review. *Digital Signal Processing*, 14:523–530.

Álvaro Gutiérrez, Campo, A., Dorigo, M., Donate, J., Monasterio-Huelin, F., and Magdalena, L. (2009). Open e-puck range & bearing miniaturized board for local communication in swarm robotics. In *IEEE International Conference on Robotics and Automation – ICRA 2009*, pages 3111–3116. IEEE Press, Piscataway, NJ.

Bagnoli, P., Brunelli, M., Magni, F., and Musumeci, D. (1976). Neural mechanisms underlying spontaneous flashing and its modulation in the firefly *Luciola lusitanica*. *Journal of Comparative Physiology*, 108:133–156.

Beni, G. (2005). *Swarm Robotics*, volume 3342/2005, chapter From Swarm Intelligence to Swarm Robotics, pages 1–9. Springer Berlin/Heidelberg.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York.

Buonomici, M. and Magni, F. (1967). Nervous control of flashing in the firefly *Luciola Italica*. *Archives italiennes de Biologie*, 105:323–338.

Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraula, G., and Bonabeau, E. (2003). *Self-Organization in Biological Systems*. Princeton University Press.

Case, J. F. and Buck, J. B. (1963). Control of flashing in fireflies 2, role of central nervous system. *Biological Bulletin*, 125:234–250.

Casper, J., Micire, M., and Murphy, R. (2000). Issues in intelligent robots for search and rescue. In *Unmanned ground vehicle technology II*, pages 292–302. SPIE, Bellingham WA, USA.

Curtis, S., Mica, J., Nuth, J., Marr, G., Rilee, M., and Bhat, M. (2000). ANTS (Autonomous Nano-Technology Swarm): An artificial intelligence approach to asteroid belt resource exploration. International Astronautical Federation, 51th Congress.

Dorigo, M. and Birattari, M. (2007). Swarm intelligence. *Scholarpedia*, 2(9):1462.

Dorigo, M. and Sahin, E. (2004). Guest editorial. Special issue: Swarm robotics. *Autonomous Robots*, 17(2–3):111–113.

Dorigo, M., Trianni, V., Groß, R., Labella, T. H., Baldassarre, G., Nolfi, S., Deneubourg, J. L., Mondada, F., Floreano, D., and Gambardella, L. M. (2004). Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots*, 17:223–245.

Holland, O. and Melhuish, C. (1997). An interactive method for controlling group size in multiple robot systems. In *Proceedings 8th International Conference on Advanced Robotics*, pages 201–206, Monterey. IEEE Press.

Holland, O., Melhuish, C., and Hoddell, S. E. J. (1997). Chorusing and controlled clustering for minimal mobile agents. In *4th Conference on Artificial Life*, pages 539–548, Cambridge. MIT Press.

Lerman, K. and Galstyan, A. (2002). Mathematical model of foraging in a group of robots: Effect of interference. *Autonomous Robots*, 13(2):127–141.

Martinoli, A. and Gambardella, L. M. (1999). A probabilistic model for understanding and comparing collective aggregation mechanisms. In *Proceedings of the 5th European Conference on Advances in Artificial Life (ECAL-99)*, volume 1674, pages 575–584. Springer.

McLurkin, J. and Yamins, D. (2005). Dynamic task assignment in robot swarms. In *Robotics: Science and Systems*, pages 129–136.

Melhuish, C., Holland, O., and Hoddell, S. E. J. (1999). Convoying: using chorusing for the formation of travelling groups of minimal agents. *Robotics and Autonomous Systems*, 28:207–216.

Mirollo, R. E. and Strogatz, S. H. (1990). Synchronization of pulse-coupled biological oscillators. *SIAM Journal on Applied Mathematics*, 50(6):1645–1662.

Mondada, F., Bonani, M., Guignard, A., Magnenat, S., Studer, C., and Floreano, D. (2005). Superlinear physical performances in a SWARM-BOT. In *Proceedings of the VIIIth European Conference on Artificial Life*, pages 282–291. Springer, Berlin, Germany.

O'Grady, R., Pinciroli, C., Christensen, A. L., and Dorigo, M. (2009). Supervised group size regulation in a heterogeneous robotic swarm. In *Proceedings of ROBOTICA 2009 - 9th International Conference on Autonomous Robot Systems and Competitions*, pages 113–119, Castelo Branco, Portugal. IPCB.

Parr, A. E. (1927). A contribution to the theoretical analysis of the schooling behaviour of fishes. *Occasional Papers Bingham Oceanographical Colloquium*, 1:1–32.

Pinciroli, C. (2006). Object retrieval by a swarm of ground based robots driven by aerial robots. Diplôme d'Etudes Approfondies en Sciences Appliquées thesis, IRIDIA, Université Libre de Bruxelles.

Pinciroli, C., O'Grady, R., Christensen, A. L., and Dorigo, M. (2009). Self-organised recruitment in a heterogeneous swarm. In *Proceedings of the 14th International Conference on Advanced Robotics (ICAR 2009)*. In press.

Sahin, E. (2005). Swarm robotics: From sources of inspiration to domains of application. In Sahin, E. and Spears, W., editors, *Swarm Robotics Workshop: State-of-the-art Survey*, number 3342 in LNCS, pages 10–20. Springer, Berlin.

Smith, H. (1935). Synchronous flashing of fireflies. *Science*, 82(2120):151–152.

Spears, W. M., Spears, D. F., Hamann, J. C., and Heil, R. (2004). Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*, 17:137–162.