International Journal on Advanced Science Engineering Information Technology

Vol.12 (2022) No. 1
ISSN: 2088-5334

Performance Evaluation of a Quorum Sensing based Scheme in Multi-Agent Task Development

Fredy Martínez [a,*], Edwar Jacinto [a], Holman Montiel [a]

[a] *Facultad Tecnológica, Universidad Distrital Francisco José de Caldas, Carrera 7 No. 40B-53, Bogotá D.C., Colombia*
Corresponding author: [*]fhmartinezs@udistrital.edu.co

***Abstract*— Robotics is positioned today as a fundamental tool in industrial and commercial development, where machines interact directly with humans. There is a vast variety of tasks that require autonomous, robust, and high-performance systems. Among these tasks can benefit from the autonomous integration of multiple elements, known as multi-agent systems. These schemes have interesting advantages over the single robot solution centered on the high degree of robustness achieved and the lower cost. The control of these multi-agent systems turns out to be of great complexity and is an active field of robotics research. The motion coordination schemes are complex and require a certain level of processing and communication. In this paper, a decentralized coordination scheme for low-cost robot groups based on local interaction is evaluated. The algorithm uses bacterial Quorum Sensing (QS) as a behavioral model, a scheme under which certain actions are triggered by the agents conditioned to the population density in the region they cover. The algorithm is tested in navigation tasks for different conditions of the design parameters. Among the parameters evaluated are environment dependence, system size, and QS threshold. The development times of the tasks were statistically analyzed, and a strong dependence of the environment on the total time required was found (a well-structured and small environment concerning the system improves the performance considerably), as well as the design of the robot in terms of QS threshold and sensors.**

***Keywords*—Autonomous systems; behavior-based control; local communication; mapping; motion; movement planning; quorum sensing; robotics; robustness; swarm.**

Manuscript received 16 Apr. 2020; revised 5 Mar. 2021; accepted 23 Apr. 2021. Date of publication 28 Feb. 2022.

I. INTRODUCTION

Industrial environments have benefited from robotic systems, particularly autonomous ones, due to the reduction in cost and increase in robustness in the development of routine tasks and those that involve some level of risk for human operators [1]. The tasks that can benefit most from a robotic application are: 1) involve transportation (moving production raw materials, moving tools in the plant, etc.), 2) require some kind of search process (identifying wet areas, hot spots, areas with too much noise pollution, or areas with some specific operational requirement), 3) require tracking and identification of animals and pests that may affect product quality or the performance of the industry [2]. It is common sense to think that autonomous artificial systems can develop these tasks since they have well-defined success criteria, do not require specialized personnel, and are routine within the production environment. A robotic system can be programmed to perform these navigation, exploration, and search tasks and even more complex tasks involving these activities. However, a single robot performing these activities has the same problem as serial systems; failure of the robot means failure to perform the task. This is why if robustness (and even lower cost) is desired, it is advisable to look for a solution based on multi-agent systems, which can guarantee the development of the task despite the partial damage of some of the elements [3]-[5]. Many low-cost platforms can be integrated within the same control algorithm for the development of a task, in principle, it is not required that the agents are identical, but that they have similar functional capabilities (similar movement, sensing, and actuation capabilities); this ensures that each agent can perform the same task.

The motion control of multi-agent systems is known for its complexity; it is well known that it is an NP-hard problem [6, 7]. When working with multi-agent or swarm robot systems, the behavior of the system is defined by the behavior of each agent, so the goal is to achieve the motion control of each robot according to the development of the task [8]. This design problem has two solution approaches. The first approach is known as odometrical, and it seeks to estimate the

position of each robot from its displacement model (derived from its structure) and the control actions applied [9]. For this strategy to work, the robot must have a very good design in terms of assembly, sensors, and response to control actions so that its mathematical model always reflects the real behavior of the platform, independently of the topology of the environment in which it moves. This also implies a knowledge of the environment that allows the robot to move correctly along regions for which its behavior is predictable [10]. In the real world, these conditions are difficult to fulfill, there are always small errors that make the behavior of the real robot differ from that expected based on the model, and this becomes much more evident in a multi-agent system, without considering the processing and communication problems that this approach implies. In real environments, it is a better approach to reactive program responses in robots. Under this approach, robots are equipped with limited sensors capable of identifying local information that allows them to decide action and movement. This prevents sensor failures (GPS, distance, etc.) from leading to the robot's deviation from its target, while at the same time reducing manufacturing costs. Also, industrial environments are highly dynamic, so a previous mapping of the environment does not yield real-time information for the robot's actions [11]. When the robot operates reactively, the important thing is to identify the neighboring region, and from this information, make decisions, which depend on whether it finds a plant worker, a rat, or a material leak [12].

The control scheme we are developing for our multi-agent systems is of the reactive type [13]. Therefore, we do not attempt to control the motion of each of our robots explicitly. Instead, we rely on the higher-level characteristics of the dynamics assigned to each agent through motion policies to generate expected behavior concerning the system. Each agent makes the motion decision individually and autonomously based on these policies and the information collected through its sensors [14]. Under this design and control principle, robots conform with a much simpler and less expensive hardware and software structure than their counterparts. They also have a specific design since they are structured to solve a particular problem (which defines their structure), and the use of high-performance hardware with multiple features that are unlikely to be used in the development of the task is avoided. The decentralized structure of the system guarantees its robustness, each agent of the system is dispensable and replaceable by another element of the system, which also reduces the complexity and capacity of the communication scheme [15]. The reactive control scheme allows simple behavioral rules to be defined from sensing. There are many examples in nature of reactive systems that have proven to be very successful in keeping organisms alive despite the conditions of the environment in which they interact. Birds, ants, and bees are some examples of multi-agent systems with their own behavioral rules that allow them to perform their activities successfully, such as feeding, surviving predators, and moving to new environments. In this sense, our research group has been developing for several years a multi-agent interaction scheme based on local interaction that mimics bacterial behavior called Quorum Sensing [16, 17]. This scheme has been successfully used in motion planning applications as well as in search algorithm optimization processes.

QS is a biological response expressed by bacteria, conditioned by genetic expressions [18]. It is a behavior identified since the 1970s that is studied in systemic biology. Bacteria read proteins deposited by themselves in the environment, and when the concentration of these proteins is high, they signal to the whole community that the population has exceeded a certain population threshold, thus transmitting to the environment a protein that allows the activation of certain behavior. This phenomenon was first observed in the bacteria *Vibrio fischeri* and *Vibrio harveyi* due to their bioluminescent response, in the former it was observed that bioluminescence only occurred when there was a certain population density [19, 20]. However, despite more than 40 years of study, the behavioral model has only been introduced in small computational applications since it had not been characterized as a common bacterial behavior.

In a simplified context and focused on path planning applications, QS can be understood as a motion control strategy that can be used on a finite group of robots to form a decentralized algorithm supported by reactive responses [21]. These reactive responses are conditioned to the local readings that each agent in the system can perform, and a collective behavior condition triggered by the population density (number of robots in the same region of the environment). Under this principle, each agent identifies specific information within the range of its sensors, including one indicating the region's population density [22]. In the natural biological model, these readings correspond to molecular concentration levels (autoinducers), but for the algorithm in robots can be replaced by something more appropriate to the application, such as some radiated signal (sound, light, the intensity of the communication signal, etc.) or information deposited in the environment (in the style of ant pheromones) that any other robot in the region can detect. Thanks to this indirect communication structure, individuals can establish the population size in their region. If this value exceeds a certain threshold, activate a specific behavior, which can indicate the location of specific elements in the environment (solution of the search problem) [23]. In bacteria, this behavior is encoded in the DNA of each bacterium; in the case of a multi-agent system composed of robots this encoding is translated into behavioral policies established in the code of each robot.

A system with an operating principle similar to the QS dynamics used in our algorithm is the voting system of ancient Greece [24]. At the time of Athenian democracy, a system was used under which, for offices that did not require specialized training, citizens were chosen at random to participate in state positions. A group of candidates based on interest and historical performance was formed, and using a roulette wheel, the selection of personnel within this group was carried out. This scheme is known as Demarchy or Stococracy and has great advantages over current selection models called conventional representative democracies, particularly because external forces do not influence the selection, and minority groups have a real possibility of participation. Like the demarchic model, our QS-based scheme, the search space is initially explored randomly, and each region has an equal chance of being visited by the group

of robots, which guarantees that in finite time the algorithm will find all possible solutions to the search problem [25]. However, the most important contribution of our model occurs later, which could be identified with the historical profile of a candidate, and that is that the best regions attract more agents. Therefore, when the population of a certain region reaches a value, it is an indicator to the system that a solution to the task has been identified. An important difference between these two models is that a chosen candidate is removed from the search space, while in our algorithm, the region remains, and its degree of the solution is weighted and compared with the other identified regions relative to the number of agents clustered in the region.

In this paper, we evaluate the performance of our QS-based reactive navigation strategy for the development of search tasks in unstructured and unknown environments, operating on the ARMOS TurtleBot 1 robotic platform [26]-[28]. In the past, this strategy has been proposed as a solution in autonomous navigation schemes, particularly in dynamic environments with direct interaction with humans (service robotics applications), and it was shown that its dynamics corresponded to a stochastic process [29, 30]. Thus, it is possible to affirm that the strategy is capable of solving search problems in this type of environment in finite time [31]. In these works, it was shown not only that the time required for the development of these tasks is finite, but also that there was a certain relationship between the total time required and the characteristics of the environment, the system, and the robot itself. As part of this research, we have conducted a statistical study with a series of laboratory tests to determine the dependencies of each of these parameters on the performance of the algorithm. These laboratory tests have been complemented with simulations to scale the behavior to large groups of robots, considering the model of the robotic platform. We have developed the ARMOS Swarm Simulator platform as an analysis tool, which contains the dynamic characteristics of our robot, and allows the programming of multiple interaction strategies over an infinite number of environment configurations.

II. Material and Method

In previous research stages, the QS-based algorithm's ability to solve simple navigation tasks was demonstrated. However, during these works, it was impossible to explicitly establish the relationship between the system parameters and the expected time to complete the tasks. In this part of the study, the efforts were focused on establishing through statistical analysis the variables that affect the system's response time and how each of these parameters affects this value. A series of experiments are performed with the ARMOS TurtleBot 1 robotic platform developing navigation tasks in controlled environments to establish these dependencies. The results are used to build a behavioral model that allows scaling the analysis to a larger number of agents. Among the parameters previously identified as key in the motion control scheme are the specific characteristics of the navigation environment, the size of the swarm population (system size), the algorithm parameters, and the capacity or range of the sensors installed on the agents (Fig. 1). The goal is to specify tasks for our system using our QS-based algorithm to guarantee the total time required for the development. This information is fundamental in real applications where robustness and total time must be guaranteed.

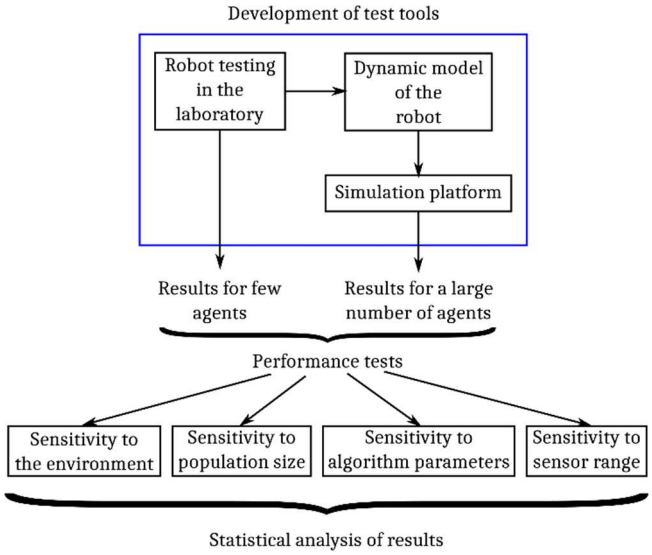

Fig. 1 Flowchart of the methodological research structure used. The ARMOS TurtleBot 1 robot is programmed with the QS-based algorithm and is subjected to behavioral tests in the laboratory. From the results, a dynamic model is built that allows scaling the behavior to large numbers of robots. These two tools are used together to evaluate the system parameters, which are then statistically analyzed to determine their actual effect.

According to the traditional notation of intelligent systems, each robot corresponds to an agent of our system. In the specialized literature, there are many movement control schemes for swarms, and it has already been noted that there are two general approaches to these schemes: specific coordination of the location of each of the agents and probabilistic coordination at the system level. In both cases, a central control unit is usually used for data processing, leading to the generation of movement policies. However, as in our case, it is increasingly common to find decentralized schemes in which data processing and movement planning processes are not performed as separate processes.

The architecture of the QS-based algorithm implements the dynamics of reactive systems. Furthermore, the system model is approached from the point of view of hybrid systems, and the specification of the tasks to be solved is performed using Linear Temporal Logic (LTL). Therefore, in the formulation of our problem, we defined a set of n autonomous agents, all identical in capacity and size, within a free space identified as E, in which the robots can move freely without physical difficulties, and which is much larger than the projection of each agent on E (that is, it is possible to accommodate in E around 100 agents without packaging problems, even though these 100 agents do not have displacement capacity).

The navigation environment W contains the free space E, which is a subset along which the agents move. For our type of robots, robots with a generalized differential model with displacement capacity on a plane in $\boldsymbol{R}^2$, W C $\boldsymbol{R}^2$ is the closure of an open set that has an open interior connected. W together with the $\boldsymbol{O}$ obstacles in its interior conforms to the navigation environment of the robots. $\boldsymbol{O}$ corresponds to the set of inaccessible regions O (O C $\boldsymbol{O}$), such that each of these regions is closed, and has a connected piecewise-analytic

boundary that, because it is contained in W, is finite. By construction, each O region is independent of each other; that is, the regions in $\boldsymbol{O}$ are pairwise disjoint, and again, finite in quantity.

The navigation environment is unknown to the agents; in fact, dynamic environments are allowed in which the shape of E and $\boldsymbol{O}$ change over time. They also do not know how many agents are in the environment. However, these agents have a limited sensor system that allows them to identify specific environmental information from which they can evaluate the quality of their current location (sensors for humidity, radiation, magnetic field, dust, etc.). They also have sensors to determine the existence and quantity of other agents in their neighboring areas. Our agent model is based on our ARMOS TurtleBot 1 robotic platform, which has nine uniformly distributed infrared sensors around its vertical axis, each with a range of 0.2 to 0.8 m (Fig. 2). These distance sensors are accompanied by an algorithm based on deep neural networks that allow each robot to identify in real-time with an accuracy margin of 93% to other agents within the range of the sensors [26, 27].

Fig. 2 ARMOS TurtleBot 1 robot. This robot is 45 cm wide and 61 cm long. Around its vertical axis, it has nine infrared distance sensors distributed along a circular aluminum bar, with 40 degrees of separation one from the other. These sensors have a range of 0.2 to 0.8 m

Sensors are essential to define the movement of each agent; through them the agents know the environment, or at least the region in which they are, the information they collect with the sensors allows them to know the environment and make decisions. These observations form an observation space S, through which the agent builds an information space I. Therefore, the agent does not have a complete map of the environment, it performs an information mapping of the form:

$$q: E \rightarrow S \quad (1)$$

We rely on the high-level dynamic properties of our strategy to guarantee agents' movement requirements. In our scheme there is no state feedback, instead each agent performs information feedback from the construction of an observation space S derived from the sensor observations made over time. These observations make up a historical observation of the form:

$$o: [0, t] \rightarrow S \quad (2)$$

The interpretation of this information space drives to the activation of movement policies that at a given moment led to the activation of specific behaviors in the agent. We say that the model of the scheme is hybrid because both the agent and the system behave continuously according to their own dynamics, but this behavior changes completely when some specific event is detected (discrete event), for example, a high population density that leads to the activation of a virulent behavior, which is characterized by a completely different dynamic.

The model contemplates two basic behaviors (two-stage process): Explorer and Virulent. The Explorer agent must navigate the environment looking for high-quality areas, while it is affected if it finds an area with a high density of agents, that is, if it detects this condition, and the number of agents exceeds a threshold of QS h, then the agent changes its behavior to Virulent (Fig. 3).

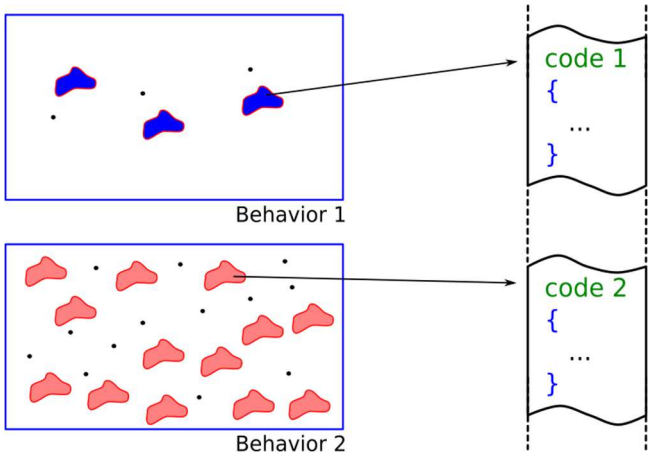

Fig. 3 The system is composed of a group of agents, each one executing a certain behavior. The proposed algorithm uses two behaviors, a first behavior leads to specific movement rules coded in a certain part of the code. If the population density is very high, the QS is activated and the second behavior is triggered, governed by a different section of code.

The activation of the virulence of an agent is done when the quorum is met. In the model both the exploring agents and the virulent agents move from one area of the environment to another according to the quality, they estimate for it, moving to areas of higher quality (according to the programmed task and the capacity of their sensors). This feature is very important in the algorithm because thanks to it each agent has the opportunity to find individually the area with the highest quality (probabilistic exploration of the state space).

These dynamic forms a parallel navigation structure that guarantees that the agents in exploratory behavior consider different areas of the environment before choosing one as a quality area. Consequently, when the quorum is reached, the agents that detect this population size characteristic begin to become virulent (Fig. 4).

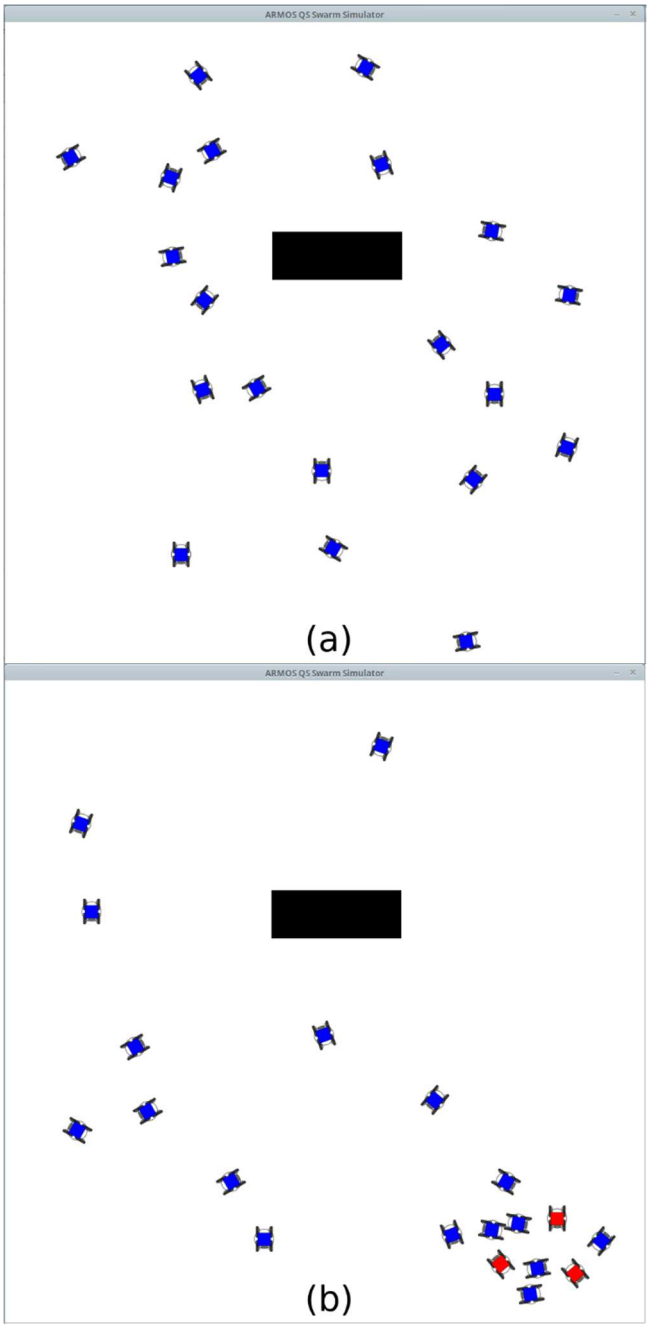

Fig. 4 ARMOS Swarm Simulator. This simulator was developed to analyze the behavior of swarms formed by the ARMOS TurtleBot 1 robot. The simulation shows a 15 m x 15 m scaled environment with an obstacle. (a) In the environment, 20 robots are placed randomly, all with initial Explorer behavior. (b) After 3 minutes, in the simulated case three robots switch to Virulent behavior (quorum threshold set to 8)

Any agent can move from one area to another if their sensors indicate that the new area is of higher quality. Fig. 5 shows the rules through which the interaction between agents and the environment occurs. As our algorithm only contemplates two behaviors, X and Z, the system's total population is constant (no reproduction of individuals). In some of the tests, the death of agents is caused, but this is equivalent to assuming damages in the system. To model the dynamics of an agent moving from one region to another within E, we assume that E is divided into finite regions (b regions in total), characterized by the same level of quality. The change from one region to another is modeled by a constant weight (μ, λ, k and ρ) defined by the quality relationship between the two regions. Therefore, there is a possibility that both explorer and virulent agents change their behavior according to their sensor readings. Because of this, the recruitment rate of agents to a region does not depend on the number of agents (QS does, but this is an additional feature of the model).

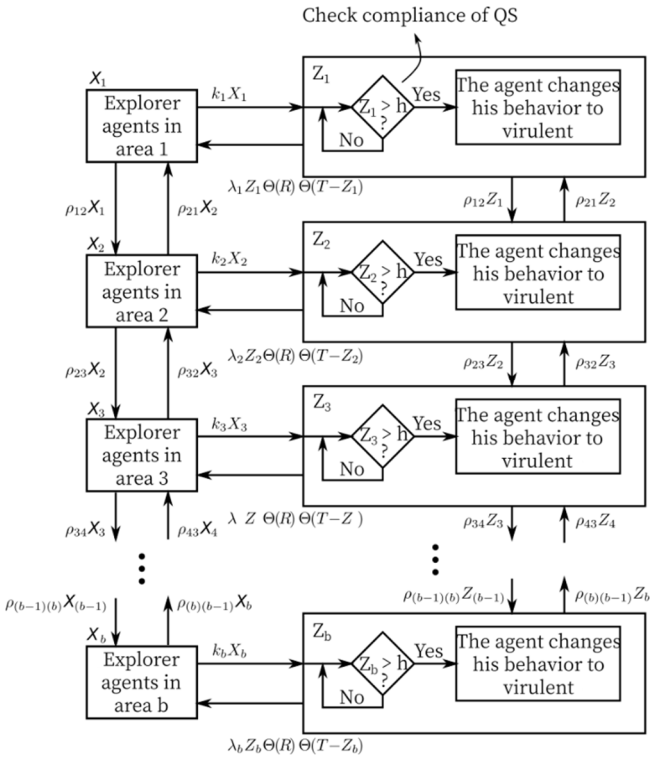

Fig. 5 Flow chart of the dynamics for the proposed QS-based algorithm

A high-quality area will naturally attract explorers. The agents move into the environment in such a way that they always try to find better regions. Eventually, the system reaches equilibrium, causing agents to concentrate on higher-quality areas, but QS accelerates this convergence since it encourages agents to stay when there are agents with virulent behavior in the region. The activation of QS is conditioned to the agent detecting a number of agents in the region higher than the threshold of quorum h. In summary, the areas with higher quality activate the quorum consensus faster.

Three elements characterize our strategy:

- Each agent is autonomous and can identify specific information in the vicinity of its location in the environment.
- The solution of the task is verified as an emergent behavior of the group, by programming the same behavior policies in each agent of the system.
- The system is infinitely scalable and robust to variations in environment, population size, and sensing capacity.

III. Results and Discussion

We took all necessary precautions to avoid bias in the results. In this sense, we ensured that parameters such as the initial position of each agent and its orientation were randomly defined, both in the tests with the real platform and

in the tests carried out by simulation. The tests were also alternated, the same parameter was never evaluated consecutively, the testing scheme was designed in such a way that the researchers did not know what type of parameter was being evaluated.

To guarantee the complete scanning of the environment, we program random movements in the robot in response to the detection of obstacles and *W* boundaries. In this way, the robots move according to the conditions of the environment and impact against obstacles, boundaries of the environment, and other robots, similar to the Brownian movement. The Brownian motion of the agents in the environment, therefore, has a strong random component. This implies that the total time of development of the task, although it is finite, is unknown, and must be evaluated statistically since it is a stochastic process. Consequently, for each possible variant that affects the task performance (size of the environment, number of agents, environment quality parameters and QS threshold) we repeat the tests 100 times, and from the results, we evaluate average behaviors and dependencies of the results.

The experiments designed for the performance evaluation were the following:

1) Experiment 1: Performance against changes in the environment: size. In this test we used a total of 20 robots with random starting position in an initial environment of 15 m × 15 m (225 m^2, Fig. 1). This size corresponded to the first test data set, for the second case the area of the environment was reduced to a size of 14 m × 14 m (196 m^2). For the remaining cases, the total area was further reduced by subtracting one meter on the vertical axis and one meter on the horizontal axis to 5 m × 5 m (25 m^2). Smaller sizes prevented the movement of the robots. In total a database of 11 cases was built, each case with a total of 100 simulations. The quorum threshold was kept constant at 8. Fig. 6 shows the results of the test with a violin plot.

2) Experiment 2: Performance against changes in the environment: quality. We code the quality of each region with a value between 0 and 1. The highest quality region is the lower right one, and in the test, we run the algorithm for 20 agents in the 15 m x 15 m environment, with a quorum threshold of 8, varying the quality of the lower right region between 0.5 and 0.9, and keeping constant the quality value of the other regions with values between 0.1 and 0.4. Again, each case was simulated 100 times. Fig. 7 shows the results of the test with a violin plot.

3) Experiment 3: Performance against changes in population size (death of agents). To evaluate the robustness of the algorithm against changes in population size, we assume that the robots may suffer damage that disables them completely, i.e., the robot may die. To evaluate this parameter, we developed several cases (each with 100 simulations) in which each one reduces the population size by one. Again, we start with our 20 agents (first case) and we reduce the population size until we reach 8 robots (this is because our quorum threshold is 8, and with a smaller population QS cannot occur). The other parameters were kept constant. Fig. 8 shows the results of the test with a violin plot.

4) Experiment 4: Performance against changes in the QS threshold. The QS in our model works as a convergence accelerator. This has been described in the formulation of the algorithm. To verify this concept, we varied the threshold value of quorum *h* from 5 to 10 in increments of 1. Fig. 9 shows the results of the test with a violin plot.

5) Experiment 5: Performance against changes in sensor range. Our intelligent distance sensor has a range of 0.2 to 0.8 m. Within this range, the robot can identify both other robots and obstacles. To evaluate the performance of the algorithm against variations of this parameter, we adjust the range of the sensor to produce cases at 0.2 m, 0.3 m, 0.4 m, 0.5 m, 0.6 m, 0.7 m, and 0.8 m. Fig. 10 shows the results of the test with a violin plot.

The free space *E* was divided into 16 regions defining four rows and four columns of equal size (they were distributed throughout *W* regardless of obstacles). The regions were assigned different quality values, but in all cases, the lower right region remained the highest quality region. In all cases, the agents were modeled according to our ARMOS TurtleBot 1 robotic platform (Fig. 2). This robot has nine infrared sensors, color sensors, an inductive sensor, independent control of its motors (it can rotate on its own axis), one DragonBoard 410C development board as control unit, and a router for Wi-Fi communication. The robot is framed in an area of 45 cm × 61 cm, and has a constant displacement speed of 2 m/s.

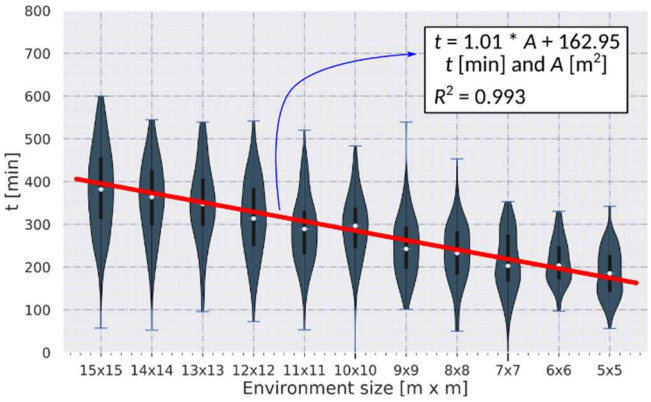

Fig. 6 Experiment 1: Task completion time (vertical) vs. Field size (horizontal)

In all the cases the data was visualized using a violin plot, which shows the detail of the median values, the probability density of the data at different values, the interquartile range, and upper and lower adjacent value. Fig. 6 shows that the time required for the development of the task increases with the increase of the navigation area, which directly implies that the algorithm is robust to these changes. The red line corresponds to the linear regression of the median values measured in each environment configuration, calculating the time in minutes versus the area of the environment measured in m^2. The result of this correlation is also shown in Fig. 6, which demonstrates that the identified growth is of linear type.

As for the quality of the environment (Fig. 7), we again observe a fairly linear dependence between the increase in quality of the region of interest and the reduction in the total time of execution of the task. Again, the algorithm proves to be robust to this parameter. Fig. 7 shows similar information as described above.

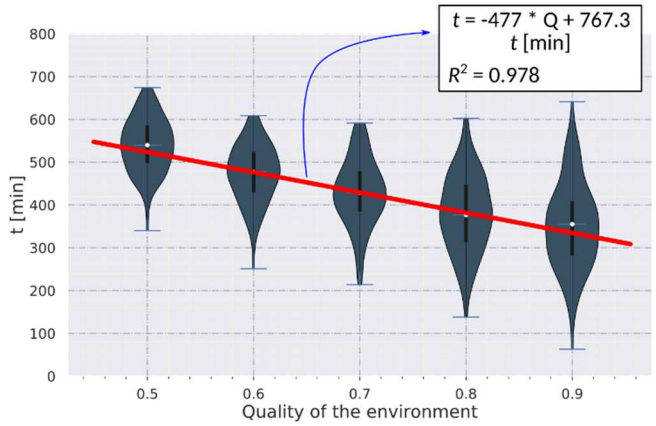

Fig. 7 Experiment 2: Task completion time (vertical) vs. Quality of the environment (horizontal)

In Fig. 8, we can see how the reduction in population size increases the time of development of the task, but the task is always completed. In this case, the relationship of time to the parameter is non-linear; it is observed that with approximately less than 12 robots, the total task development time is almost tripled. Furthermore, it is observed that there is no major impact on increasing the number of robots above 14. This behavior shows that the algorithm is also robust to the population size of the system.

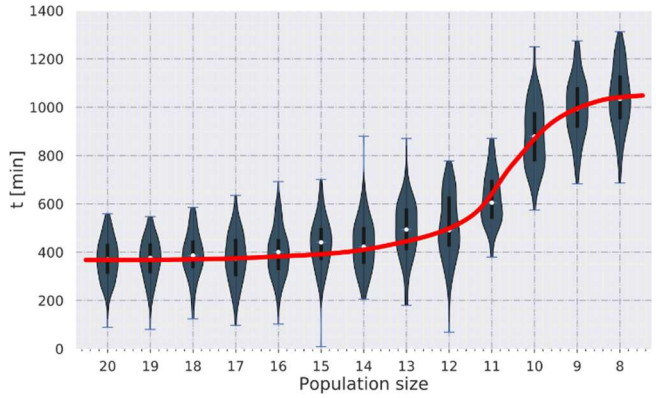

Fig. 8 Experiment 3: Task completion time (vertical) vs. Population size (horizontal)

Again, there is a linear relationship between the parameter and the total time of the task. Fig. 9 shows how the time decreases as the quorum threshold increases and that at least within the values evaluated, this relationship is linear, according to the equation shown in the figure.

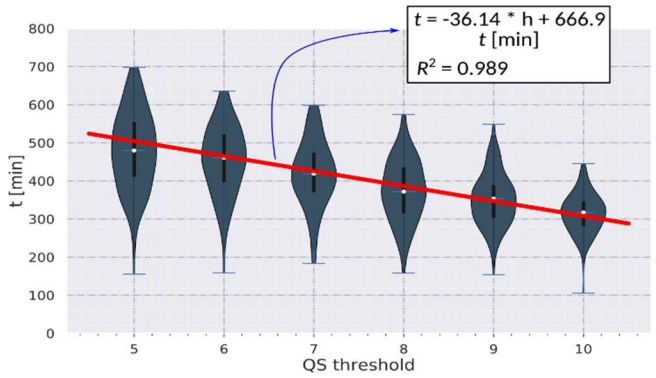

Fig. 9 Experiment 4: Task completion time (vertical) vs. QS threshold (horizontal)

Finally, Fig. 10 shows a relationship of reducing the total task time by increasing the sensing range and that this relationship is not linear. This is to be expected since the robot cannot identify the other robots with very little range until it is very close to them, making QS difficult. When increasing the sensor range above 0.4 m, the reduction in time behaves in an almost linear way. In these cases, the exploration and QS are more important for the convergence of the algorithm, and these parameters are not affected by the capacity of the sensors. According to these results, the algorithm is proven to be robust to sensor performance changes.

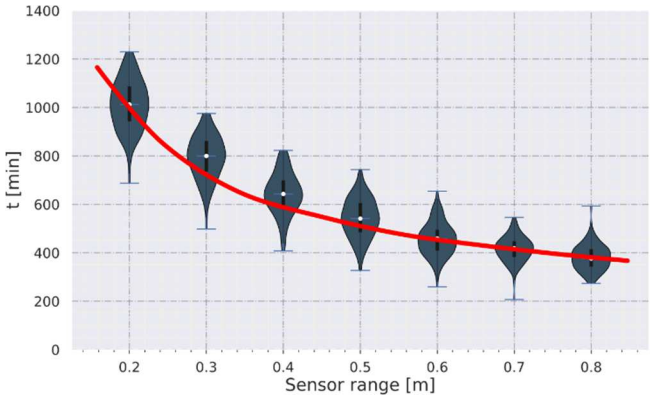

Fig. 10 Experiment 5: Task completion time (vertical) vs. Sensor range (horizontal)

The results confirm the behavior reported in previous research while providing formal expressions for establishing task development times under certain conditions. While it is not possible to explicitly define the development time, the curves and their behavior allow us to obtain expressions for the expected time and population size relationships concerning the size of the environment and the time required to develop a task. They also establish behavioral relationships about sensor capacity and QS threshold, marginally reducing task time within certain ranges. The performance of our control scheme against other centralized control approaches implies the definition of basic points of comparison, additional to the total time required by the task, such as robustness against agent failure, percentage of success in the development of the task, and costs of the solution. In general, our approach far exceeds the robustness and reliability of centrally controlled schemes, but such a comparison is outside the focus of this research.

IV. Conclusion

The research group is developing its applications in hardware, software, and control strategies in service robotics. Previous work has shown the ability of its reactive algorithm based on feedback behavior to solve navigation tasks in dynamic and unknown environments. The major restriction of the strategy is the impossibility of formally defining the task development time as a function of the task parameters. This paper seeks to statistically solve this issue by evaluating the performance of the algorithm on a multi-agent system consisting of the ARMOS TurtleBot 1 robotic platform concerning task design parameters such as the size of the navigation environment, the configuration of the environment quality as the basic robot grouping mechanism, the size of the

system population, the value of the quorum threshold used, and the ability of the sensors to detect their neighboring robots. The objective of the studied tasks is to group the swarm in the regions of interest of the environment according to the task and sensors in the robots. For this purpose, we propose using a navigation algorithm inspired by bacterial interaction that includes QS-activated behavior. Under this model, the robots navigate the environment trying to advance towards the highest quality regions while monitoring the number of robots in their neighboring region. The algorithm converges by allowing the robots to cluster in the areas they identify as being of the highest quality, a process that is accelerated if the population size exceeds a threshold. We perform simulations from the dynamic model of our ARMOS TurtleBot 1 robot platform, hoping that the results can be generalized into reality for work with this robot. The experiments were designed considering random distribution and navigation of the robots, the reason why each case study was supported with 100 simulations. The results showed that the QS-based motion control algorithm is robust to changing the parameters considered and that this behavior can be modeled and predicted. Large environments require larger systems. The total task time depends linearly on the environment but is strongly reduced when the population size increases. The latter relationship is not linear, and as observed, depends on the size ratio between these two parameters (population size concerning environment size). Very large populations do not significantly benefit the strategy, but very small populations do reduce performance considerably. According to the results, the ideal ratio for our ARMOS TurtleBot 1 is one robot per 15 m^2.

NOMENCLATURE

E	Free space
h	Threshold of QS
I	Information space
n	Number of agents in the system
S	Observation space
O	Obstacle
$\boldsymbol{O}$	Set of all obstacles
W	Navigation environment
X	Explorers' population
Z	Virulent population

Greek letters

μ, λ, k and ρ Constants that weight easy of passage between regions

ACKNOWLEDGMENT

This work was supported by the Universidad Distrital Francisco José de Caldas, in part through CIDC, and partly by the Facultad Tecnológica. Universidad Distrital does not necessarily endorse the views expressed in this paper. The authors thank the research group ARMOS for the evaluation carried out on prototypes of ideas and strategies.

REFERENCES

[1] H. ElGibreen and K. Youcef, "Dynamic task allocation in an uncertain environment with heterogeneous multi-agents," *Autonomous Robots*, vol. 43, no. 7, pp. 1639–1664, 2019.

[2] S. Krivic and J. Piater, "Pushing corridors for delivering unknown objects with a mobile robot," *Autonomous Robots*, vol. 43, no. 6, pp. 1435–1452, 2019.

[3] F. Berlinger, J. Dusek, M. Gauci, and R. Nagpal, "Robust maneuverability of a miniature, low-cost underwater robot using multiple fin actuation," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 140-147, 2018.

[4] Z. Liu, C. West, B. Lennox, and F. Arvin, "Local bearing estimation for a swarm of low-cost miniature robots," Sensors (Switzerland), vol. 20, no. 11, pp. 1-23, 2020.

[5] T. Xuehong, L. Huanlao, and L. Haitao, "Robust finite-time consensus control for multi-agent systems with disturbances and unknown velocities," *ISA Transactions*, vol. 80, no. 1, pp. 73-80, 2018.

[6] Z. Jing, Z. Xiaozhe, Z. Xiaopan, Z. Dongdong, and L. Huanhuan, "Task Allocation for Multi-Agent Systems Based on Distributed Many-Objective Evolutionary Algorithm and Greedy Algorithm," *IEEE Access*, vol. 8, no. 1, pp. 19306-19318, 2020.

[7] W. Guang, X. Ming, W. Yiming, Z. Ning, X. Jian, and Q. Tong, "Using Machine Learning for Determining Network Robustness of Multi-Agent Systems Under Attacks," *Lecture Notes in Computer Science*, vol. 11013, no. 1, pp. 491-498, 2018.

[8] A. Nazarova, and M. Zhai, "Distributed Solution of Problems in Multi Agent Robotic Systems," *Studies in Systems, Decision and Control*, vol. 174, no. 1, pp. 107-124, 2019.

[9] Y. Jiang, H. Yedidsion, S. Zhang, G. Sharon, and P. Stone, "Multi-robot planning with conflicts and synergies," *Autonomous Robots*, vol. 43, no. 8, pp. 2011–2032, 2019.

[10] Z. Hashemifar, C. Adhivarahan, A. Balakrishnan, and K. Dantu, "Augmenting visual slam with wi-fi sensing for indoor applications," *Autonomous Robots*, vol. 43, no. 8, pp. 2245–2260, 2019.

[11] O. Saha, P. Dasgupta, and B. Woosley, "Real-time robot path planning from simple to complex obstacle patterns via transfer learning of options," *Autonomous Robots*, vol. 43, no. 8, pp. 2071–2093, 2019.

[12] G. Ferrer and A. Sanfeliu, "Anticipative kinodynamic planning: multi-objective robot navigation in urban and dynamic environments," *Autonomous Robots*, vol. 43, no. 6, pp. 1473–1488, 2019.

[13] A. Rendón, "Evaluation of autonomous navigation strategy based on reactive behavior for mobile robotic platforms," *Tekhnê*, vol. 12, no. 5, pp. 75-82, 2015.

[14] A. Hock and A. Schoelling, "Distributed iterative learning control for multi-agent systems," *Autonomous Robots*, vol. 43, no. 8, pp. 1989–2010, 2019.

[15] G. Li, D. Onge, C. Pinciroli, A. Gasparri, E. Garone, and G. Beltrame, "Decentralized progressive shape formation with robot swarms," *Autonomous Robots*, vol. 43, no. 6, pp. 1505–1521, 2019.

[16] T. Rijavec, J. Zrimec, R. Spanning, and A. Lapanje, "Natural microbial communities can be manipulated by artificially constructed biofilms," *Advanced Science*, vol. 6, no. 22, pp. 1–12, 2019.

[17] M. Schuster, D. Sexton, and B. Hense, "Why quorum sensing controls private goods," *Frontiers in Microbiology*, vol. 8, no. 885, pp. 1–16, 2017.

[18] S. Mukherjee, and B. Bassler, "Bacterial quorum sensing in complex and dynamically changing environments," *Nature Review Microbiology*, vol. 17, no. 1, pp. 371–382, 2019.

[19] S. McAnulty and S. Spencer, "The role of hemocytes in the hawaiian bobtail squid, euprymna scolopes: A model organism for studying beneficial host-microbe interactions," *Frontiers in Microbiology*, vol. 7, no. 2013, pp. 1–8, 2017.

[20] L. Tanet, C. Tamburini, C. Baumas, M. Garel, G. Simon, and L. Casalot, "Bacterial bioluminescence: Light emission in photobacterium phosphoreum is not under quorum-sensing control," *Frontiers in Microbiology*, vol. 10, no. 365, pp. 1–9, 2019.

[21] N. Tabassum, "Quorum Sensing-A Communication Pathway for Behavioural Synchronization in Bacteria," *International Journal of Medical Studies*, vol. 4, no. 1, pp. 7-11, 2019.

[22] A. Rasouli, P. Lanillos, G. Cheng, and J. Tsotsos, "Attention-based active visual search for mobile robots," *Autonomous Robots*, vol. 44, no. 2, pp. 131–146, 2020.

[23] J. Postat and P. Bousso, "Quorum sensing by monocyte-derived populations," *Frontiers in Immunology*, vol. 10, no. 2140, pp. 1–7, 2019.

[24] P. Cartledge, Cultures of Voting in Pre-modern Europe. Taylor & Francis Group, 1 ed., 2018.

[25] B. Ichter, and M. Pavone, "Robot Motion Planning in Learned Latent Spaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407-2414, 2019.

[26] F. Martínez, A. Rendón, and M. Arbulú, "A data-driven path planner for small autonomous robots using deep regression models," *Lecture Notes in Computer Science*, vol. 10943, no. 1, pp. 596–603, 2018.
[27] F. Martínez, E. Jacinto, and H. Montiel, "Neuronal environmental pattern recognizer: Optical-by-distance LSTM model for recognition of navigation patterns in unknown environments," *Communications in Computer and Information Science*, vol. 1071, no. 1, pp. 220–227, 2019.
[28] J. Caley, N. Lawrance, and G. Hollinger, "Deep learning of structured environments for robot search," *Autonomous Robots*, vol. 43, no. 7, pp. 1695–1714, 2019.
[29] M. Castiblanco, and F. Martínez, "Exploración de un modelo comportamental basado en el Quorum Sensing bacterial para describir la interacción entre individuos," *Tekhnê*, vol. 11, no. 1, pp. 21-26, 2014.
[30] D. Ezzat, S. Amin, H. Shedeed, and M. Tolba, "A New Nano-robots Control Strategy for Killing Cancer Cells Using Quorum Sensing Technique and Directed Particle Swarm Optimization Algorithm," *Advances in Intelligent Systems and Computing*, vol. 921, no. 1, pp. 218-226, 2019.
[31] G. Cai, and D. Sofge, "An Urgency Dependent Quorum Sensing Algorithm for N-SiteSelection in Autonomous Swarms," *18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019)*, pp. 1853-1855, 2019.