2011 IEEE International Conference on
Automation Science and Engineering
Trieste, Italy - August 24-27, 2011

Integrating human swarm interaction in a distributed robotic control system

Cristian Vasile and Ana Pavel and Cătălin Buiu
Laboratory of Natural Computing and Robotics
Politehnica University of Bucharest
Splaiul Independenţei, no. 313,
060042 Bucharest, Romania
{cvasile, apavel, cbuiu}@ics.pub.ro

Abstract— **A multi-agent control architecture for swarm robotics applications which includes an innovative human-swarm interface is proposed. The architecture is designed to allow an operator to monitor and guide a robotic swarm to accomplish its missions. The system is composed of three types of agents, a graphical user interface agent, and a pair of a local and a social agent for each robot in the swarm. The local agent implements low level robot-specific functionalities like movement, obstacle avoidance and localization. The control algorithm is implemented in the social agent and is based on an adapted distributed version of the Particle Swarm Optimization technique. An original method, Gravity Points Method, for representing goals which are used by the human-swarm interface is also proposed. Experimental results using simulated e-puck robots are presented and directions for further developments are given.**

I. INTRODUCTION

Real-world environments are dynamic and unstructured. Typical tasks for collective robotic systems working in such environments include searching for targets and transportation of various loads. A multi-agent based control architecture which addresses these kind of missions is proposed. This architecture is named Chidori which literally means "One Thousand Birds" in Japanese [1], in order to suggest its primary application area, that is , swarm robotics. Each robot does not posses cognition, but only a reactive behavior. On the other hand, the swarm as an entity is a intelligent system due to its emergent behavior. Chidori implementation is designed as a multi-agent system in order to be scalable, flexible and extensible. A human-swarm interface was also integrated in the architecture to allow a human expert to guide the multi-robot system to accomplish the tasks in such a way which does not limit the intelligent emergent behavior of the swarm.

An adapted distributed Particle Swarm Optimization (PSO) inspired algorithm is used in the control architecture. Previous work of the authors include experiments with a PSO inspired algorithm using Khepera III robots [2]. Chidori architecture was presented in an earlier designing stage in [1] where the technologies involved in the implementation were also presented. Related work is presented in [3] where the authors describe an adapted PSO inspired algorithm for robotic swarms and discuss the differences between abstract particles and robots and the effects of physical laws and properties on PSO algorithm. A distributed PSO version is presented in [4]. Another approach to swarm control is physicomimetics (artificial physics), which uses virtual forces to achieve desired configurations ([5]). The states of the swarm are computed by minimizing the virtual system potential energy. This framework was also used to integrate human-swarm interaction [6]. The proposed Chidori architecture is designed to be a distributed control architecture that supports human-swarm interaction. Besides the structure of the architecture itself, the main contribution of the paper is the Gravity Points Method (GPM) described in Sect. III-B which was designed to allow an operator to guide a swarm through a swarm-user interface. The GPM method is a general technique to define goals and can be well integrated with the PSO technique.

II. CHIDORI ARCHITECTURE

A. Overview

The proposed cognitive control architecture (Fig. 1) for a swarm of robots is designed as a multi-agent system. Three types of agents are defined in order to provide a distributed control system and a human-swarm interface. Each robot in the swarm is mapped to a pair of agents, a local agent and a social agent. These agents implement the behavior of the robot when interacting with the environment (local behavior), and with the other robots (social behavior). The third type of agent, graphical user interface (GUI) agent, is used by human operators to interact with the swarm.

The proposed multi-agent architecture is designed to be scalable, so, adding a robot to the swarm corresponds to adding a new pair of agents (local and social) to the distributed control system. It is also designed to be modular and extensible as new functionalities can be added to the agents. Fault tolerance is achieved by the fact that disabled robots won't cause the system's failure and by a self-healing mechanism described in Section IV.

The proposed architecture is a hybrid one as it has both a reactive component and a hierarchical structure (Fig. 2). The reactive behavior, defined by the local agent, is given by the interaction with the environment (for example object avoidance, target detection) and by the commands received

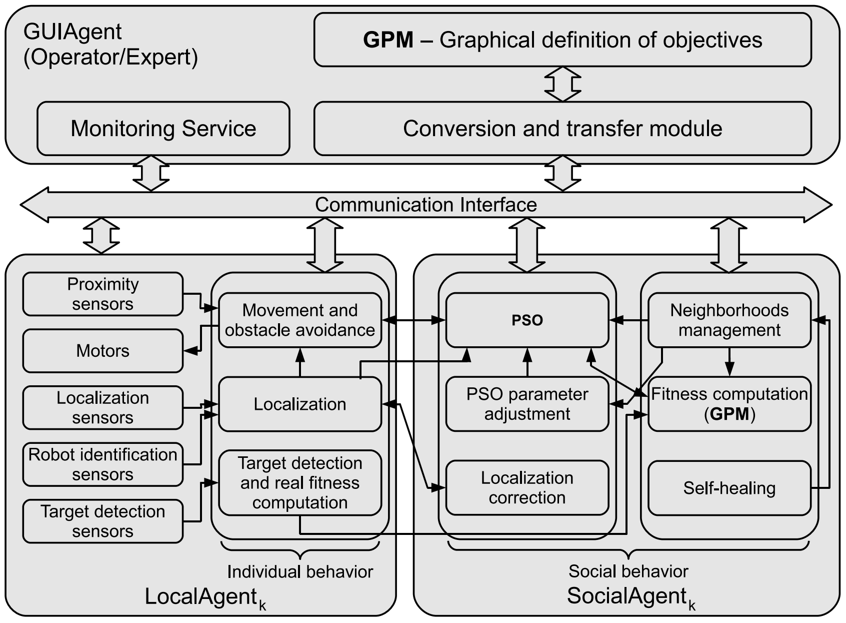

Fig. 1. Chidori multi-agent control arhitecture and human-swarm interface – the pair of agents corresponding to the *k*-th robot and the GUI agent which has only one instance in the multi-agent system

from the corresponding social agent. Social interaction between robots is the key component in a swarm which allows solving problems and generates the emergent behavior of the swarm. This social interaction generates a hierarchical structure in the swarm. Two types of groups of robots were defined in order to model the swarm hierarchy: sub-swarms and neighborhoods (Fig. 2). Sub-swarms are groups of robots in the swarm which try to achieve different goals. For example, sub-swarm 2 may search for a target, while sub-swarm 6 transports an object or searches for another target, see Fig. 2. Sub-swarms themselves are composed of one or more neighborhoods. Neighborhoods belonging to the same sub-swarm are smaller groups of robots which help each other in achieving the common goal of the sub-swarm. For example, when searching for a target, the robots may group themselves in neighborhoods in order to help each other pass local minima and reach the target's location (neighborhoods 3, 4, and 5 in Fig. 2). Another example is the transportation of an object by groups of robots which are formed in order to carry and balance it)neighborhoods 7, 8, 9, and 10 in Fig. 2). Each robot belongs to one and only one neighborhood and therefore to one and only one sub-swarm. The swarm has a dynamic hierarchical structure due to the changes in the goal list, to the available robots and to other factors.

Another important aspect addressed by the proposed architecture is the problem of interfacing human operators with a robotic swarm. This type of interface must be scalable with the number of robots in the swarm, flexible in order to be efficient when the structure of the swarm changes, extensible and expressive in order to allow the user to add, change or remove the swarm's goals. The operator cannot and should not control each robot in the swarm and therefore the interface must offer easy and accessible ways to guide the swarm and to display the swarm's state in a relevant and expressive way. The emergent intelligent behavior of the multi-robot system must not be limited by the operator. Guidance is used to help the swarm achieve its goals. For example, the human expert may have information about dangerous or interesting areas in the search space of a mission. A technique to define, represent and transmit such knowledge to the swarm is proposed in Section III.

B. *Local Agent*

The local agent is responsible for the individual behavior of a robot. Because it interacts directly with the hardware, the local agent is specific for each type of robot. It can be implemented both for simulated and real robots. The only requirement is that the robot, real or simulated, must have a minimum set of sensors and actuators to implement its functionalities. The required set of hardware resources is composed of:

- proximity sensors, used for obstacle avoidance (example: infrared, ultrasonic, laser range finder);
- localization sensors (example: encoders, GPS);
- target detection sensors (example: microphone, infrared, camera);
- robot detection sensors, used for avoiding other robots and for localization correction (example: infrared);
- motors for movement (differential wheeled type, rover type, legged type).

Some hardware may be used for more than one functionality.

The local agent's functionalities are implemented in three modules: *Movement and obstacle avoidance module*, *Localization module* and *Target detection and real fitness compu-*

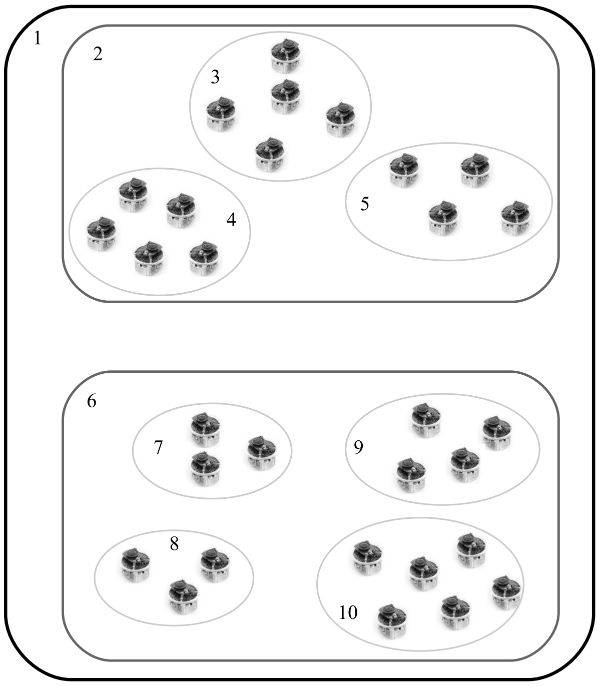

Fig. 2. The hierarchical structure of Chidori.Border 1 encloses the whole swarm in which two sub-swarms have formed, delimited by borders 2 and 6. In each sub-swarm a number of neighborhoods can be noticed (the light gray borders 3, 4, 5 in the first sub-swarm and the light gray borders 7, 8, 9, 10 in the second sub-swarm)

tation module, as shown in Fig. 1. Implementation details are presented in Section IV.

C. Social Agent

The social agent defines the interaction between robots and implements the distributed control system. It generates commands for its corresponding local agent, based on the goals of the robot's neighborhood and sub-swarm. All functionalities offered by this agent use the inter-agent communication and are not dependent on the robot type. The social agent's implementation is structured in six modules divided in two classes: three low-level modules and three high-level modules.

1) Low-level modules: PSO module, *PSO parameter adjustment module* and *Localization correction module*. The *PSO module* uses a distributed Particle Swarm Optimization (PSO) inspired algorithm to generate the next position of the robot which is sent to the corresponding local agent. The control algorithm uses the current location of the robot obtained from the local agent and the fitness value computed based on the neighborhood's goals. The PSO parameters can be adjusted in order to fine tune its performance (*PSO parameter adjustment module*). In order to maintain the algorithm's performance the *Localization correction module* was developed to generate correction values for the robot's position, based on physical interaction with other robots. Negotiation is employed to compute the positioning error correction of the involved robots.

2) High-level modules: Neighborhoods management module, *Fitness computation module* and *Self-healing module*. The *Neighborhoods management module* (NM) is responsible for allocating the swarm's resources, different types of robots, to sub-swarms and neighborhoods. This resource planning operation is done according to the expert's goals and preferences. The preferences may include goals' priorities, the number and type of robots allocated to the goals, but are only taken in consideration as suggestions. The configuration of the swarm's structure is generated by the distributed NM module. Goals are defined by the operator (external goals) and by the swarm (internal goals). NM assigns these goals to the sub-swarms which are then transformed into fitness values by the *Fitness computation module*. When searching for targets, fitness computation also uses data received from the *Target detection and real fitness computation module*. Because of limited target detection ranges, the operator estimates the position of the target and guides the swarm. When the target is detected, the estimations are replaced by the real position of the target. The proposed method used to allow the expert to guide the swarm is the Gravity Points Method (GPM) which is described in detail in Sect. III. The swarm's internal goals are defined by the *Self-healing module*. A robot that needs help (for example when stuck in a hole) has to make other robots aware of its state so they can come and help. This can be accomplished by generating an internal goal for the swarm, thus the robot which needs help becomes a target.

D. GUI Agent

The interface between the swarm and the human operator is implemented in the GUI agent. There is only one instance of this agent in the multi-agent system which is responsible for offering graphical tools for goal definition by using the GPM method (see Section III) and swarm's state visualization. The operator uses the interface to guide and monitor the swarm in order to complete the mission. These functionalities are implemented in three modules: *Graphical definition of objectives module*, *Conversion and transfer module* and *Monitoring service module*. The *Conversion and transfer module* converts objectives from a graphical format into a numerical one which is understood by the swarm. The *Monitoring service module* displays the state of a robot, a group of robots or the entire swarm, in an useful and expressive way for the operator.

III. Chidori implementation techniques

The proposed architecture is implemented around two important techniques: Particle Swarm Optimization (PSO) and Gravity Points Method (GPM). Although the Chidori architecture is designed to be general as it does not depend on a certain implementation or techniques, the two proposed methods are used in the implementation because they can be easily integrated together in the system.

A. Particle Swarm Optimization (PSO)

An adapted distributed PSO inspired algorithm is used in the implementation of the swarm control system. PSO is a global stochastic optimization method inspired by the social behavior of bird flocks, fish schools and insect societies. It is a population based algorithm developed by J. Kennedy and R. Eberhart [7]. The individuals in the swarm are named particles which fly through the search space of the problem in order to find the global optimum. The solution can be represented as points in a multi-dimensional search space and it defines the fitness function that is optimized. The PSO algorithm is based on the interaction between particles which form neighborhoods in order to pass local optima. Each particle has three parameters: velocity, position and local optimum. The velocity is computed as follows:

$$\begin{aligned} v_i(k+1) &= w * v_i(k) + r_1 * c_1 * (p(k) - x_i(k)) \\ &\quad + r_2 * c_2 * (n(k) - x_i(k)) \\ &\quad + r_3 * c_3 * (g(k) - x_i(k)) \qquad (1) \\ x_i(k+1) &= x_i(k) + v_i(k+1) \\ &\quad x_i, v_i, p, n, g \in R^n \end{aligned}$$

where n is the dimension of the search space, v_i and x_i are the *i*-th particle's velocity and position, p, n and g are the local, neighborhood and global optima, w is the particle's inertia, c_1, c_2, c_3 are the learning factors corresponding to the local, neighborhood and global component, r_1, r_2, r_3 are random variables uniformly distributed in $[0, 1]$.

It is shown in [7] that grouping particles in neighborhoods increases the PSO algorithm's search power, but lowers its convergence speed. Because of this property sub-swarms and neighborhoods are defined in the Chidori architecture in order to increase the chance of finding the target. The components of the PSO algorithm (global, neighborhood, local) are mapped to a sub-swarm, the sub-swarm's neighborhoods and the corresponding robots. Sub-swarms are working independently to achieve different goals, having different instances of the PSO algorithm. Each robot in a sub-swarm has a corresponding particle used in the distributed PSO instance of its sub-swarm.

The adapted PSO-inspired algorithm is implemented in a distributed way. This property raises the problems of communication and synchronization of the control algorithm. The number of neighborhoods of a sub-swarm determines the amount of communication used for negotiation of the neighborhoods' optima and the global optimum. In each neighborhood a robot (negotiator) is designated to negotiate the global optimum with the other neighborhoods and broadcast it to the robots in its neighborhood. The process of selecting a negotiator is fault tolerant because it can use any of the available robots from the neighborhood and the negotiator can be dynamically changed in any moment of the sub-swarm's mission. Therefore, when the negotiator robot becomes disabled, another robot will take its place. The implementation of this mechanism is discussed in Section IV. Designing a distributed PSO algorithm must take into account the difference between abstract particles and real robots which have physical properties like mass, dimension, orientation and are subject to the physical laws [3].

B. Gravity Points Method (GPM)

GPM is an original method proposed by the authors which is used for representing goals. This representation is used to implement the Human-Swarm Interface. The method is used both for defining goals in a graphical manner and then transforming them into a numerical format understandable by the swarm, and for defining multi-purpose fitness functions used by the PSO algorithm.

Goals can be either external, given by the human expert through the GUI agent, or internal, generated by the swarm using the self-healing mechanism. In GPM, goals are represented as virtual attractive and repulsive points. Virtual attractive points are used to mark interesting areas or to estimate a target position. They can also be used by the swarm to mark the position of a robot that needs help. On the other hand, repulsive points are used to indicate uninteresting or hazardous areas. This method is used only to guide the swarm and must not be used to control the robots. It is not a planning method like other potential field techniques [8] because the swarm's movement is determined by the PSO algorithm. Therefore the operator needs to define only position estimates of a target in a search mission and not routes defined by check points. The PSO algorithm is a powerful stochastic search technique which is able to autonomously search the areas around virtual points in order to detect real targets. This emergent behavior should not be limited by the operator.

Real targets cannot always be perceived by sensors because of their limited target detection ranges (Fig. 3). In this case, an operator can only estimate the target's position using knowledge and previous experience. When the target is detected, the virtual points are replaced by the position of the real target.

The swarm uses the GPM method to compute the fitness function. Attractive points contribute to the optimum of the fitness function, while the repulsive points have a negative contribution. The fitness function can be defined in many ways, a simple form being the following:

$$F(x) = \sum_{a \in AP_S} f(x, a) - \sum_{r \in RP_S} f(x, r) \qquad (2)$$

where F is the fitness function, x is the current position of a robot, f is the contribution function of x and an attractive or repulsive point, which may depend on the distance between them, AP_S and RP_S are the sets of attractive and repulsive points in the sub-swarm S. The contribution function, f, can be chosen as $f(x, p) = G(\|x - p\|)$, where G is the Gaussian function and p is either an attractive point, a, or a repulsive point, r. The parameters of the Gaussian function can be used to shape and fine tune the potential field generated by an attrative or repulsive point. The fitness function, F, is maximized by the PSO algorithm. The robots, which are in fact the particles, try to get to the attractive points while avoiding the repulsive ones.

The GPM technique is a global method for interacting with the swarm. As far as the authors know, other research groups have treated different aspects of human-swarm interfacing.

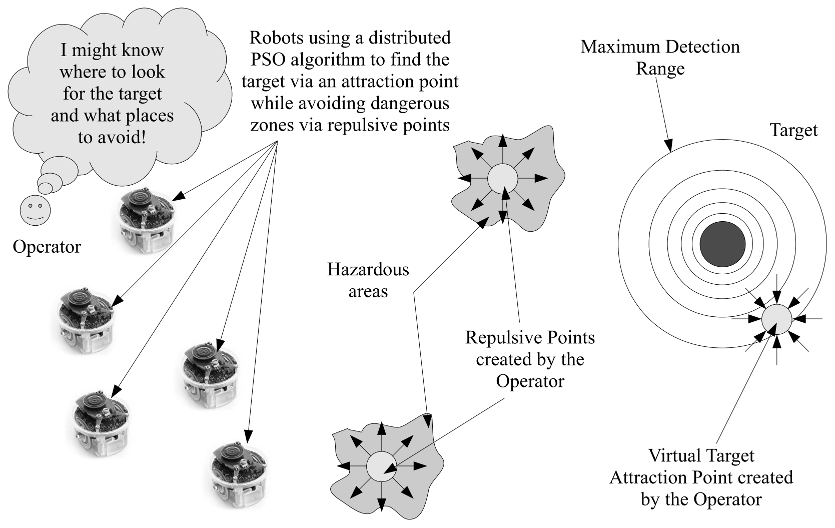
I might know where to look for the target and what places to avoid!
Operator
Robots using a distributed PSO algorithm to find the target via an attraction point while avoiding dangerous zones via repulsive points
Hazardous areas
Repulsive Points created by the Operator
Maximum Detection Range
Target
Virtual Target Attraction Point created by the Operator

Fig. 3. Operator guided search for a target while avoiding hazardous areas

One approach uses visual and acoustic signals to express the swarm's state in order to develop an infrastructure for programming and debugging swarm applications [9]. Another approach uses a local method to interface with a swarm which implements PSO as a control algorithm [10]. The user controls an avatar which is a virtual instance of a robot. The other robots consider the user-controlled avatar as being one of them. The operator tries to use the avatar to find the target helping the other robots in their search. The PSO uses the fitness values computed for the avatar to determine the global optimum. The proposed architecture differs from the first approach which is focused on swarm applications development and debugging and from the second approach which uses a robot control method to help the swarm.

IV. Software implementation of Chidori architecture

The multi-agent control system is implemented in Java, based on the Java Agent DEvelopment Framework, JADE. The JADE software platform is used in Chidori implementation because it offers an inter-agent communication interface, an API for creating the agents and defining their behaviors and a run-time platform for agent based applications.

The main architectural elements of the JADE platform are agents and containers. JADE agents are implemented as threads and run in containers which are Java processes. Containers offer agent hosting and execution services. There is a special container named *Main Container* which is the first container created in the platform initiation operation. All the other containers must register to the Main Container. The Main Container manages the containers table, the agents' descriptor table and two special agents: agent management service (AMS) and yellow pages service (DF). JADE has a special mechanism to replicate the Main Container in order to make the multi-agent application fault-tolerant. The replication mechanism which is not active by default is enabled in Chidori implementation.

Inter-agent communication is based on ontologies and semantic languages. Semantic languages are used to represent knowledge in a platform independent way. Ontologies represent the common vocabulary used by the agents and they are used to define the concepts, predicates and agent-actions, belonging to a specific domain. JADE agents communicate knowledge by using the Content Manager class which transforms Java objects into a semantic language (SL, Leap or XML) based on the registered ontology. More details about JADE framework can be found in [11]. The ontology used by Chidori agents is defined using the Protégé framework.

The functionalities of the three agents defined in the proposed multi-agent system, *LocalAgent*, *SocialAgent* and *GUIAgent*, are implemented as behaviors. Behaviors contain the code which is executed by the JADE agent. Four behaviors were registered in the LocalAgent. The first one is an update behavior which is executed periodically in order to exchange data between the robot and its model. The robot model is a data structure which is used to store sensor data and commands which are sent to the robot. The other three behaviors are the movement behavior which is responsible for moving the robot to a given location and avoiding obstacles, localization behavior which is responsible for computing the location of the robot based on odometry and detecting other robots, and the target detection behavior which is used to compute a fitness value based on sensor data received from the real target. Infrared sensors are used both for obstacle avoidance and for other robots detection and recognition. Motor encoders are used for localization and microphones, for the target detection.

SocialAgent uses three behaviors, PSO behavior which implements fitness computation, parameters' adjustment and particles' positions computation, NM behavior which is responsible for managing the sub-swarms and their neighborhoods and also for the self-healing mechanism, and the localization correction behavior which uses data from encountered robots to correct their position. The self-healing mechanism is integrated in the NM behavior because both internal and external goals are processed in the same way.

GUIAgent offers two services implemented in two behaviors. The first behavior is the monitoring service behavior which is designed to retrieve and visualize the state of the swarm at four levels: robot level, neighborhood level, subswarm level and swarm level. The second behavior is the graphical user input interface which uses the GPM method to define goals as virtual attractive and repulsive points in a graphical manner. It is also responsible for transforming the goals into a numerical format and transmitting them to the swarm using the JADE communication interface (Fig. 1). GUI is developed using JavaFX technology which is easy to use and integrate with Java based applications.

Webots simulator is used for fast development and debugging of the implementation. It is a professional robotics simulator which offers various models for robots, sensors and actuators. Webots was interfaced with the local agents using a custom TCP/IP protocol. The simulator also offers the possibility of sending commands directly to real robots instead of the simulated ones. This is a very useful feature because no source code must be created or changed in order to test the implementation on the real robots.

V. Simulation results

Chidori architecture was implemented as a multi-agent system based on JADE Framework. The implementation was tested on simulated e-puck robots (7cm diameter) in the Webots 6 simulation environment. The experiments were conducted in a flat rectangular arena (3x2.1m size) as shown in Fig. 4. The tests prove that the local agent's functionalities, movement, obstacle avoidance and odometric localization, are working properly together with the PSO control algorithm implemented in the social agent. A JavaFX script was also successfully interfaced with the GUIAgent. The communication between agents, Webots and real e-puck robots was also tested and is well functioning.

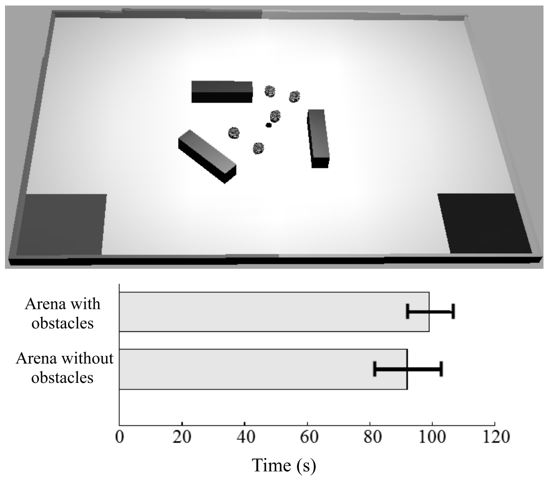

Fig. 4. Swarm of robots searching for a target (black point in the center of the arena)

In all the simulated tests, the robots were able to find a given virtual attractive point, marked as a black circle in the center of the arena (Fig. 4). The chart in Fig. 4 shows the average execution time and standard deviation in an arena with and without objects. The average time to find the virtual point is 92s for the arena without objects and 99s for the arena with obstacles. The robots were initially scattered randomly near the arena's boundaries. Fig. 4 shows the final configuration of 5 e-puck robots which converged to the virtual attractive point. The experimental results are promising and are used to guide further development of the Chidori implementation and test cases design on real robots.

VI. Conclusions and future work

An integrated multi-robot control architecture was proposed. together with a human-swarm interface. The implementation of the multi-agent system and preliminary test results were discussed. The authors proved that the proposed architecture can be used to develop swarm applications.

Further work includes improvements of the local and social behaviors. Also, the graphical component will be further developed to achieve the presented requirements.

New experiments are being designed and conducted in order to test and validate the architecture and the human-swarm interface. This will include test cases using both simulated and real robots.

VII. Acknowledgments

This work was supported by CNCSIS UEFISCSU, project number PNII IDEI 1692/2008.

References

[1] C. Vasile, A. Pavel, and C. Buiu, "Chidori - a bio-inspired cognitive architecture for collective robotics applications," *IFAC Workshop on Intelligent Control Systems WICS2010*, pp. 52–57, 2010.

[2] C. Vasile and C. Buiu, "A software system for collaborative robotics applications and its application in particle swarm optimization implementations," *Applied Soft Computing*, p. submitted, 2010.

[3] J. Pugh and A. Martinoli, "Inspiring and modeling multi-robot search with particle swarm optimization," *IEEE Swarm Intelligence Symposium*, pp. 332–339, 2007.

[4] ——, "Distributed adaptation in multi-robot search using particle swarm optimization," *10th International Conference on the Simulation of Adaptive Behavior, Lecture Notes in Computer Science*, pp. 393–402, 2008.

[5] W. M. Spears, D. F. Spears, and D. Zarzhitsky, "Physicomimetics positioning methodology for distributed autonomous systems," *In Proc. of the Government Microcircuit Applications and Critical Technology Conference (GOMACTech-05)*, 2005.

[6] Z. Kira and M. Potter, "Exerting human control over decentralized robot swarms," *4th International Conference on Autonomous Robots and Agents*, 2009.

[7] J. Kennedy, R. Eberhart, and Y. Shi, *Swarm intelligence*. San Francisco: Morgan Kaufmann, 2001.

[8] Y. Hwang and N. Ahuja, "A potential field approach to path planning," *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 23–32, 1992.

[9] J. McLurkin, J. Smith, J. Frankel, D. Sotkowitz, D. Blau, and B. Schmidt, "Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots," *AAAI Spring Symposium*, 2006.

[10] S. Bashyal and G. K. Venayagamoorthy, "Human swarm interaction for radiation source search and localization," *IEEE Swarm Intelligence Symposium*, pp. 21–23, 2008.

[11] F. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. West Sussex: Wiley, 2007.