Robotics and Autonomous Systems 62 (2014) 200–213

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Benchmark of swarm robotics distributed techniques in a search task

Micael S. Couceiro [a,b,*], Patricia A. Vargas [c], Rui P. Rocha [a], Nuno M.F. Ferreira [b]

[a] Institute of Systems and Robotics, Department of Electrical and Computer Engineering, University of Coimbra, Pólo II, 3030-290 Coimbra, Portugal
[b] RoboCorp, Electrical Engineering Department, Engineering Institute of Coimbra, Rua Pedro Nunes - Quinta da Nora, 3030-199 Coimbra, Portugal
[c] Robotics Laboratory, School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, United Kingdom

HIGHLIGHTS

- A survey on multi-robot search inspired on swarm intelligence is presented.
- Five state-of-the-art swarm robotic algorithms are described and compared.
- Simulated experiments of a mapping task are carried out to compare the five algorithms.
- The three best performing algorithms are deeply compared using 14 e-pucks on a source localization problem.
- The Robotic Darwinian Particle Swarm Optimization (RDPSO) algorithm depicts an improved convergence.

ARTICLE INFO

Article history:
Received 12 July 2013
Received in revised form
11 October 2013
Accepted 25 October 2013
Available online 7 November 2013

Keywords:
Swarm robotics
Search tasks
Benchmark
Performance analysis

ABSTRACT

This paper presents a survey on multi-robot search inspired by swarm intelligence by further classifying and discussing the theoretical advantages and disadvantages of the existing studies. Subsequently, the most attractive techniques are evaluated and compared by highlighting their most relevant features. This is motivated by the gradual growth of swarm robotics solutions in situations where conventional search cannot find a satisfactory solution. For instance, exhaustive multi-robot search techniques, such as sweeping the environment, allow for a better avoidance of local solutions but require too much time to find the optimal one. Moreover, such techniques tend to fail in finding targets within dynamic and unstructured environments. This paper presents experiments conducted to benchmark five state-of-the-art algorithms for cooperative exploration tasks. The simulated experimental results show the superiority of the previously presented Robotic Darwinian Particle Swarm Optimization (*RDPSO*), evidencing that sociobiological inspiration is useful to meet the challenges of robotic applications that can be described as optimization problems (e.g., search and rescue). Moreover, the *RDPSO* is further compared with the best performing algorithms within a population of 14 e-pucks. It is observed that the *RDPSO* algorithm converges to the optimal solution faster and more accurately than the other approaches without significantly increasing the computational demand, memory and communication complexity.

1. Introduction

Search applications have been well studied in the past [1]. However, the use of multi-robot systems (*MRS*) to fulfill such missions has not yet received the proper attention. Nonetheless, *MRS* offer several advantages over single solutions, or even human rescuers, within search applications. Besides providing a natural fault-tolerance mechanism, the use of multiple robots is especially preferable when the area is either hazardous or inaccessible to humans, e.g., search-and-rescue (*SaR*) victims in catastrophic scenarios [2].

Similar to optimization problems in which one can distinguish exhaustive methods from biologically-inspired ones, *MRS* within search applications face the same dilemma: either decide on an exhaustive technique in which robots sweep the entire area [3], or mimic simple local control rules of several biological societies

* Correspondence to: RoboCorp, Department of Electrical Engineering, Engineering Institute of Coimbra (ISEC), Rua Pedro Nunes, 3031-601 Coimbra, Portugal. Tel.: +351 239 790 382.
E-mail addresses: micaelcouceiro@isr.uc.pt, micael@isec.pt (M.S. Couceiro).

0921-8890/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.robot.2013.10.004

(e.g., ants, bees, birds) to stochastically search the scenario [4]. This last one is a typical feature from most *swarm robotics* algorithms [5]. Swarm robotics applied in search tasks can offer several major benefits over the conventional search techniques, such as: the robustness of the swarm to failure of individual units or run-time addition of new units, the scalability of emergent behaviors to swarms of different sizes, the leveraging of self-organization principles of environmental noise and individual differences, and the synergetic effect whereby the work of the swarm is greater than the sum of the work by the individual units, known as *superlinearity* [4]—a concept shared with other fields such as complex systems.

Although many different swarm robotics algorithms have been recently proposed in the literature, this work will focus on the ones that benefit from explicit over implicit communication, such as the work recently proposed by Kernbach et al. [6]. In algorithms under explicit communication, robots need to be able to explicitly exchange information within a network path using some sort of medium (e.g., wireless communication). Despite such a requirement, the choice of explicit communication over implicit or *through-the-world* interactions, also known as *stigmergy*, relies on the application domain of realistic search applications such as search-and-rescue (*SaR*). According to the current state-of-the-art in this field, robotic technology is used, almost exclusively, to assist and not substitute human responders. Hence, multiple mobile robots can take advantage of parallelism to reduce the time required to fulfill the mission while explicitly providing important data about the site (e.g., contextual information), whether accessible or inaccessible for human agents.

Moreover, this work will focus on distributed solutions. This is obvious within swarm robotics context in which tasks are inherently distributed in space, time, or functionality. Nevertheless, it should be noted that some works still emphasize on centralized architectures [7], thus moving away from the fully distributed nature inherent to the principles of collective intelligence. In practice, centralized swarm architectures are computationally expensive and unsuitable as a large number of robots usually generate very dynamic behaviors that a centralized controller cannot handle [8]. Also, centralized architectures lack robustness as the failure of the centralized entity may compromise the performance of the whole *MRS* [9].

Bearing these ideas in mind, this work presents an overview of distributed swarm robotics techniques under explicit communication constraints, applied to search applications, thus comparing them in both simulated environment and real experiments.

2. Swarm robotics in search applications

In nature, some complex group behaviors arise in biological systems composed of swarms that are observed in a variety of simple social organisms (e.g., ants, bees) [10]. One of the most relevant topics in *MRS* is the modeling and control of the population. Hence, the design of such bio-inspired swarm *MRS* requires the analysis of the social characteristics and behaviors of insects and animals.

To that end, Suarez and Murphy [2] recently presented a brief description of more than 50 papers on animal foraging making the analogy to *SaR* applications. Most works presented in this survey suggested that robots should divide the whole environment into patches, as many animals do, and then search within such patches. Nevertheless, and even as stated by the authors, victims can appear anywhere. Hence, the difficulty in subdividing a search environment and defining patches within unknown scenarios still remains. The authors also claim that *SaR* robotics should focus on exhaustive search as the motivation is different from animal foraging—while animals attempt to maximize their net energy level to stay alive, robots must find victims in a search area or determine that there are none to be found. However, although optimization may seem unsuitable for *SaR* robotics at first, there are some specific applications in which one can foresee its use like, for instance, in urban fires. Urban fires are probably the most frequent catastrophic incidents in urban areas, requiring a prompt response because of life endangerment in highly populated zones and the high risk of fire propagation to buildings and parked cars in the vicinity. An urban fire in a large basement garage often frequented by people and containing inflammable materials, like in a basement garage of a shopping mall with many cars, drivers and people passing by, is a particularly challenging *SaR* application because of the confined nature of the environment. As the fire evolves, the space becomes rapidly full of smoke, with very poor visibility and an unbreathable and toxic atmosphere, which is dangerous for both victims and first responders. Moreover, victims prone to such atmosphere would be unable to survive more than 10 to 20 min. Therefore, the use of exhaustive search strategies within this context would be unfeasible.

Actually, some works have recently focused on such scenarios, such as *Cooperation between Human and rObotic teams in catastroPhic Incidents* (*CHOPIN*) *R&D Project* [11]. The *CHOPIN* project aims at exploiting the human–robot symbiosis in the development of human rescuers' support systems for *SaR* missions in urban catastrophic incidents. One of the main catastrophic scenarios being used for proof of concept is the occurrence of fire outbreaks in large basement garages. In this case, the project focuses on using a fleet of cooperative ground mobile robots to cooperatively explore a basement garage where the fire is progressing, thus identifying the localization of fire outbreaks and victims.

One of the first approaches to fulfilling the objectives of this project was in dividing that kind of application into two operations: (i) *reconnaissance*; and (ii) *rescuing* [12]. In both phases, this kind of scenario usually poses radio propagation difficulties to the response teams, whose members usually wear a radio emitter/transmitter to communicate by voice. Often under these noisy scenarios, communication is only possible with teammates located in line-of-sight. Moreover, since a wireless communication computer network may be absent or damaged, robotic agents may have to deploy and maintain a mobile ad hoc wireless network (*MANET*) in order to support the interaction between the human team and the robotic team. In the *reconnaissance* phase, the mission consisted of a team of robots that arrived at the scenario via a common entry and spread out to explore and map the unknown area, signalizing possible points of interests such as victims and fire outbreaks. After a certain degree of confidence in the built representation of the scenario, the *rescuing* phase consisted of having the team of robots inspecting the area in a coordinated way, visiting all points of interest, looking for remaining victims, possible changes in the scenario and the evolution of fire outbreaks. The first approach was handled using a complete solution based on the well-known*Particle Swarm Optimization* (*PSO*) [13] to real mobile robots, denoted as *Robotic Darwinian PSO* (*RDPSO*), that was previously presented in [14] and further extended in [15,16].

Due to the successive improvements of the *RDPSO* and its positive outcome on several search tasks, now comes the time to benchmark it with state-of-the-art alternatives. Over the past few years, some algorithms initially designed to solve tasks such

as optimization problems have been adapted to embrace the principles associated to real robots. Within that list, and including the aforementioned *RDPSO*, the following ones were found as the most promising for realistic search task applications:

(1) Robotic Darwinian Particle Swarm Optimization (*RDPSO*) [14,15].
(2) Extended Particle Swarm Optimization (*EPSO*) [17,18].
(3) Physically-embedded Particle Swarm Optimization (*PPSO*) [19,20].
(4) Glowworm Swarm Optimization (*GSO*) [21,22].
(5) Aggregations of Foraging Swarm (*AFS*) [23,24].

Note that three-out-of-five algorithms gain inspiration from the *PSO*, proposed for the first time by Kennedy and Eberhart in 1995 [13]. This socio-inspired algorithm takes advantage of the swarm intelligence concept defining the properties of a system of unsophisticated agents, locally interacting with their environment, whose behavior creates coherent global functional patterns [25]. Given its simplicity in terms of implementation, and reduced computational and memory complexities, the *PSO* has been successfully used in many applications such as robotics [26–29], computer vision [30], electric systems [31] and social sciences [32]. Although the *PSO* has been mainly used on optimization and estimation problems, many recent studies have been adapting it to follow swarm robotics principles. As such, this work presents and compares three of the most significant studies around *PSO*-based swarm robotic algorithms.

The next section briefly describes the *RDPSO* algorithm previously presented by the authors. Afterwards, the *RDPSO* will be systematically compared and discussed over the alternative swarm robotics algorithms.

2.1. Robotic Darwinian Particle Swarm Optimization (RDPSO)

The *RDPSO* initially proposed by Couceiro, Rocha and Ferreira [14,15], just like the *PSO*, basically consists of a population of robots that collectively move in the search space (e.g., catastrophic scenario, city) in search of the optimal solution (e.g., number of victims, number of passengers); each robot is characterized by its pose (i.e., position and heading) and performance. For instance, if we have a group of mobile olfactory robots that are trying to find a gas leak in an indoor environment (*cf.*, [33,34]), each robot's state comprises of its pose and the corresponding value of gas density.

The *RDPSO* summarized in Algorithm 1 allows multiple dynamic swarms, thus enabling a distributed approach, because the network that might have been comprised of the whole swarm of robots is divided into multiple smaller networks (one for each group/swarm). This makes it possible to decrease the number of nodes (i.e., robots) and the information exchanged between robots of the same network. In other words, robots' interaction with other robots through communication is confined to local interactions inside the same group (swarm), thus making *RDPSO* scalable to large populations of robots. More details about this work will be highlighted throughout this paper for comparison purposes with the other strategies.

Although this is not the first work extending the *PSO* to *MRS*, a more recent work by Couceiro et al. has shown that the *RDPSO* can overcome problems related to obstacle avoidance, robot dynamics, sub-optimal solutions and communication constraints (e.g., [35–37]).

Algorithm 1. *RDPSO* algorithm for robot n.

Initialize pose $\langle x_n[0], \varphi_n[0]\rangle$ and $swarmID$ based on EST
Loop:
  If $swarmID \neq 0$ // it is not an excluded robot
    Evaluate the robot individual solution $h_n[t]$
    If $h_n[t] > h_{best}$ // robot has improved
      $h_{best} = h_n[t]$ // individual best sensed solution
      $\chi_1[t] = x_n[t]$ // individual best position
    Exchange information with the N_s teammates about the individual solution $h_n[t]$ and current position $x_n[t]$
    Build a vector $H[t]$ containing the individual solution of all N_s robots within $swarmID$
    If $\max H[t] > H_{best}$ // swarm has improved
      $[H_{best}, j] = \max H[t]$ // j will return the best robot of $swarmID$
      $\chi_2[t] = x_i[t]$ // swarm's best position
      If $SC_s > 0$
        $SC_s = SC_s - 1$ // stagnancy counter
      If $SC_s = 0$ // the swarm can be rewarded
        $probability\_of\_calling\_new\_robot()$ or
        $probability\_of\_creating\_new\_swarm()$
    Else // swarm has not improved
      $SC_s = SC_s + 1$ // stagnancy counter
      If $SC_s = SC_{max}$ // punish swarm
        $exclude\_worst\_performing\_robot()$
    If $g_n[t] \geq g_{best}$ // maximize distance to obstacles
      $g_{best} = g_n[t]$
      $\chi_3[t] = x_n[t]$
    $\chi_4[t] = enforce\_communication()$
    $[\alpha, \rho_1, \rho_2, \rho_3, \rho_4] = fuzzyfied\_contextual\_info()$
    $w_n[t] = -\sum_{k=0}^{r} \frac{(-1)^k \Gamma(\alpha+1) v_n[t+1-k]}{\Gamma(k+1)\Gamma(\alpha-k+1)}$
    $v_n[t+1] = w_n[t] + \sum_{i=1}^{4} \rho_i r_i (\chi_i[t] - x_n[t])$
    $x_n[t+1] = x_n[t] + v_n[t+1]$
  Else // it is an excluded robot
    $wandering\_algorithm()$
    Evaluate its individual solution $h_n[t]$
    Exchange information with the N_X teammates about the individual solution $h_n[t]$ and current position $x_n[t]$
    Build a vector $H[t]$ containing the individual solution of all N_X robots within the excluded swarm ($swarmID = 0$)
    If $\max H[t] > H_{best}$
      $H_{best} = \max H[t]$
      If $h_{best} = \max_{N_I} H[t]$ // this is one of the best N_I performing robot of the excluded swarm
        $probability\_of\_creating\_new\_swarm()$
        If receives information about the need of a new robot
          $swarmID = swarmID\_received$ // include this robot in the active swarm
          $N_S = N_S + 1$
          Exchange information with teammates about N_S
        If receives information about the need of creating a new swarm
          $swarmID = swarmID\_new$ // include this robot in a new active swarm
          $N_S = N_I$ // reset number of robots in the swarm
          $N_s^{kill} = 0$ // reset number of excluded robots
          $SC_s = 0$ // reset search counter
until stopping criteria (convergence/time)

2.2. Extended Particle Swarm Optimization (EPSO)

In fact, one of the first adapted versions of the *PSO* to handle real world constraints, such as obstacles, was presented by Pugh and Martinoli [17,18] (Algorithm 2). The main difference between the algorithm presented by those authors, denoted hereafter as *Extended PSO* (*EPSO*), and the classical *PSO* is that each robot (or particle) only takes into consideration the information of the robots within a fixed radius r_c (omnidirectional communication). Hence, contrarily to the *RDPSO* [15], the *EPSO* algorithm does not use multi-hop connectivity and does not constrain robots' motion so as to ensure some degree of communication network connectedness.

Moreover, and also contrarily to the *RDPSO* [14] algorithm in which obstacle avoidance behavior is integrated in the main equations of robots' motion, the authors used the *Braitenberg* obstacle avoidance algorithm [38]. Hence, if a robot is executing a step of the algorithm and avoids an obstacle, it will continue moving in its new direction but will not modify its internal velocity representation. Although such a methodology makes it possible to decouple the high level behavior of robots from collision avoidance routines, such a strategy may be unfeasible if one needs to study the stability of the algorithm considering obstacles influence over robots [36], or even define adaptive methodologies to systematically adjust all the algorithm parameters based on contextual information [16].

Algorithm 2. *EPSO* algorithm for robot n.

Initialize pose $\langle x_n[0], \varphi_n[0]\rangle$ randomly defined
Loop:
  Evaluate the robot individual solution $h_n[t]$
  If $h_n[t] > h_{best}$ // robot has improved
    $h_{best} = h_n[t]$
    $\chi_1[t] = x_n[t]$
  Exchange information with the N_s neighbors about the individual solution $h_n[t]$ and current position $x_n[t]$
  Build a vector $H[t]$ containing the individual solution of all N_s robots within a fixed radius r_c
  If $\max H[t] > H_{best}$ // swarm has improved
    $[H_{best}, j] = \max H[t]$ // j will return the best neighbor
    $\chi_2[t] = x_j[t]$
  $v_n[t+1] = wv_n[t] + \sum_{i=1}^{2} \rho_i r_i(\chi_i[t] - x_n[t])$
  $x_n[t+1] = x_n[t] + v_n[t+1]$
  $x_n[t+1] = Braitenberg\_obst\_avoid(x_n[t+1])$
until stopping criteria (convergence/time)

Pugh and Martinoli [17,18] evaluated the performance of their learning technique for a simple task for robot groups of various sizes. The authors analyzed how the performance of the standard *PSO* neighborhood structure was affected by adapting it to a more realistic model, which considers limited communication abilities. Experimental results obtained using the *Webots* simulator [39] showed that the adapted version of the *PSO* maintained good performance for groups of robots of various sizes when compared to other bio-inspired methods such as Genetic Algorithms. However, contrarily to the presented *RDPSO* algorithm [14], all bio-inspired methods used in this work, including the adapted *PSO*, tend to get trapped in sub-optimal solutions, i.e., the authors do not present any strategy to avoid sub-optimal solutions.

2.3. Physically-embedded Particle Swarm Optimization (PPSO)

Similarly, Hereford and Siebold [19,20] presented a *Physically-embedded PSO* (*PPSO*) in swarm platforms (Algorithm 3). As in *RDPSO* [14], there is no central agent to coordinate the robots movements or actions. The authors constrained the movement of particles within a limited cone to avoid the omnidirectionality inherent to the common *PSO*. Although this strategy seems practical, this could be achieved by considering the dynamical characteristics of robots. For instance, the *RDPSO* [37] benefits from fractional calculus of order α to avoid drastic changes in a robot's direction.

Algorithm 3. *PPSO* algorithm for robot n.

Initialize pose $\langle x_n[0], \varphi_n[0]\rangle$ randomly defined
Loop:
  Evaluate the robot individual solution $h_n[t]$
  If $h_n[t] > h_{best}$ // robot has improved
    $h_{best} = h_n[t]$
    $\chi_1[t] = x_n[t]$
  Exchange information with the N_s neighbors about the individual solution $h_n[t]$
  Build a vector $H[t]$ containing the individual solution of all N_s robots within a fixed radius r_c
  If $\max H[t] == h_n[t]$ // it is the best robot
    Exchange information of the current position $x_n[t]$ if it is the best performing robot within a fixed radius r_c
  If $\max H[t] > H_{best}$ // swarm has improved
    $[H_{best}, j] = \max H[t]$ // j will return the best neighbor
    $\chi_2[t] = x_j[t]$
  $v_n[t+1] = wv_n[t] + \sum_{i=1}^{2} \rho_i r_i(\chi_i[t] - x_n[t])$
  $x_n[t+1] = x_n[t] + v_n[t+1]$
  $x_n[t+1] = constrain\_movement(x_n[t+1])$
  while $collision() == 1$
    $x_n[t+1] = go\_back\_turn\_right(x_n[t+1])$
until stopping criteria (convergence/time)

The algorithm presented by Hereford and Siebold [19,20] also assumed the synchronization of robots, such that robots would only compute a novel position after all other robots exchange the necessary information (e.g., individual solutions). Also, robots would only share their position if their own solution is the best solution in the whole swarm. This makes it possible to reduce the amount of communication traffic, however, it also requires that robots would stop after each iteration in order to handle all relevant information. This is an interesting strategy when using broadcasting mechanisms since robots can share information among themselves without requiring lots of communication traffic. Nevertheless, such a strategy would not significantly improve the algorithm performance if the team would benefit from ad hoc communication with multi-hop properties.

Despite the potentialities of the physically-embedded *PSO* presented by Hereford and Siebold [19,20], experimental results were carried out using a population of only three robots, performing a distributed search in a scenario without sub-optimal solutions. Also, although the authors present experimental results with one and two obstacles, the collision avoidance behavior was not considered within the algorithm's equation. Instead, once a robot got stuck or collided, it was programmed to go back and turn right.

2.4. Glowworm swarm optimization

A distributed biologically algorithm inspired by glowworm behavior was presented and applied in *MRS* by Krishnanand and Ghose [21,22] (Algorithm 4). Similarly to the *RDPSO*, the *Glowworm*

Swarm Optimization (*GSO*) algorithm features an adaptive decision domain which enables the formation of subgroups in the population where the goal is to partition the population of robots to track multiple sources concurrently. Nevertheless, and contrarily to the *RDPSO* that uses a set of fuzzy rules and performance evaluation of both robots and swarms of robots [14], the *GSO* acts more like a *PSO* with best neighborhood solution information. In fact, to begin a search, a robot chooses a neighbor to be its leader and moves towards it. The most probable choice for the leader is the one with the highest *luciferin* value, i.e., the luminescence quantity that represents the individual solution, thus corresponding to the most probable direction of the source. As a result of this leader selection, subgroups form within the population and begin searching for nearby solutions. In other words, as no evolutionary techniques are used, it is shown that all members of a single cluster will converge to the leader at some finite time, and members of overlapping clusters will converge to one of the leaders asymptotically.

Similarly to Pugh and Martinoli [17,18] and Hereford and Siebold [19,20], the authors also incorporated a low-level obstacle avoidance model, thus allowing robots to turn away from detected obstacles to prevent collisions.

Unfortunately, and despite the algorithm potentialities, the experiments were carried out using only four wheeled physical robots and a target location using a single sound source.

2.5. Aggregations of foraging swarm

Another interesting approach was presented by Gazi and Passino [23,24] in which the swarm is modeled based on attractant/repellent profiles as aggregations of foraging swarm (*AFS*) (Algorithm 5). These kinds of attractant/repellent profiles are consistent with biological observations [40], where the inter-individual attraction/repulsion is based on an interplay between attraction and repulsion forces with the attraction dominating on large distances, and the repulsion dominating on short distances. As in the *RDPSO* [36], the authors presented a stability and convergence analysis of their algorithm. To that end, the authors carried out a behavioral analysis followed by several simulation experiments so as to define the most adequate values of the system parameters.

Algorithm 4. *GSO* algorithm for robot n.

Initialize pose $\langle x_n[0], \varphi_n[0]\rangle$ and luciferin level $l_n[0]$ randomly defined
Loop:
- $l_n[t] = (1-\rho)l_n[t-1] + \gamma h_n[t]$ // update the luciferin level
- $N_n = \{j: d_{nj}[t] < r_d^n[t];\ l_n[t] < l_j[t]\}$ //determine neighbors of glowworm n in the local-decision range and with higher luciferin levels
- Exchange information with the N_n selected neighbors about the individual solution $l_n[t]$ and current position $x_n[t]$
- $L_n[t] = \frac{l_j[t]-l_n[t]}{\sum_{k\in N_n} l_k[t]-l_n[t]}$ // calculate probability of selecting neighbor j from the N_s neighbors
- $[L_{best}, j] = \max L[t]$ // j will return the best neighbor glowworm
- $x_n[t+1] = x_n[t] + s\left(\frac{x_j[t]-x_n[t]}{\|x_j[t]-x_n[t]\|}\right)$ // move toward neighbor j
- $r_d^n[t+1] = \min\{r_c, \max\{0, r_d^n[t] + \beta(\eta_t - |N_n|)\}\}$ // update local-decision range based on specified number of neighbors

until stopping criteria (convergence/time)

This is worth mentioning since most of the works define the parameters using a trial-and-error mechanism. Hence, some sort of mathematical formalism, such as stability analysis, is required to enable the obtaining of such a comparable performance.

Despite this, the authors did not present any mechanism for sub-optimal solutions avoidance. Therefore, the convergence of the swarm cannot be proved in the general case, thus demonstrating the difficulty of obtaining general guarantees for progress properties.

In this approach, the authors consider obstacles as a part of the objective function of the swarm. In other words, if robots need to maximize a given measure (e.g., find the larger density of victims in a catastrophic incident), obstacles are considered as global minima of their objective function. This is not too different from the *RDPSO* case that benefits from another component to define obstacles, i.e., a monotonic and positive *sensing function* that depends on the sensing information [14]. Nevertheless, the approach presented by Gazi and Passino [23,24] does not make it possible to adjust robots' behavior depending on the presence or absence of obstacles. Put differently, the swarm behavior is limited to convergence in the vicinity of a solution or divergence from the neighborhood of a sensed obstacle, being unable to adapt to the adequate contextual information.

Algorithm 5. *AFS* algorithm for robot n.

Initialize pose $\langle x_n[0], \varphi_n[0]\rangle$ randomly defined
Loop:
- Evaluate the robot individual solution $h_n[t]$ and distance to obstacles $g_n[t]$
- Exchange information with the N_T robots from the population about the individual solution $h_n[t]$ and current position $x_n[t]$
- $\sigma_n[t] = \gamma g_n[t] - \rho h_n[t]$ // build the attractant/repellent "σ-profile" of attractant substances (main mission objective) and repellent substances (obstacles)
- $J(x_n[t]) = \sum_{k\in N_T} -(x_n[t] - x_k[t])\left[a - be^{-\frac{\|x_n[t]-x_k[t]\|^2}{c}}\right]$
- $v_n[t+1] = -\nabla\sigma_n[t] + J(x_n[t])$ // compute the velocity of robot n based on the information of all N_T robots from the population and its own "σ-profile"
- $x_n[t+1] = x_n[t] + v_n[t+1]$

until stopping criteria (convergence/time)

Although the work of Gazi and Passino [23,24] does not assume any specifications about communication constraints, their model controls agents individually but each agent needs to know the positions of all other agents in the swarm. Therefore, we will consider that this approach requires multi-hop communication and the same principles assumed for the *RDPSO* will be considered.

2.6. Summary

For theoretical comparison purposes, a summary of the previously presented algorithms is presented in Table 1, thus highlighting the most pertinent features for *MRS* applications. An empty cell in the table indicates that the algorithm does not benefit from that feature or there is no pertinent information in the literature to support it.

Robot dynamics consists of constraining agents' dynamics to fulfill the requirements inherent to the limited mobility of robots. From the previously presented works, only two consider this feature. The *PPSO* presented a simple rule to constrain robots' movements within a limited cone, while the *RDPSO* uses fractional

Table 1
Summary of swarm algorithms for search tasks.

	RDPSO [14,15]	*EPSO* [17,18]	*PPSO* [19,20]	*GSO* [21,22]	*AFS* [23,24]
Robot dynamics	Fractional calculus		Constrained movements		
Obstacle avoidance	Artificial repulsion	Low-level control	Low-level control	Low-level control	Artificial repulsion
Initial deployment	*EST* approach	Random	Random	Random	Random
Communication	Ad hoc multi-hop	Broadcast	Broadcast	Broadcast	Ad hoc multi-hop
Fault-tolerance	Multi-connectivity				
Parameterization	Stability analysis				Stability analysis
Avoid sub-optima	Punish–reward mechanism based on natural selection				
Multiple and dynamic sources	Dynamic partitioning & fuzzy adaptive behavior			Partitioning	
Computational complexity	$\mathcal{O}(2N_S)$	$\mathcal{O}(N_S)$	$\mathcal{O}(N_S)$	$\mathcal{O}(N_S)$	$\mathcal{O}(N_T)$
Memory complexity	$\mathcal{O}(r_\alpha)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Communication complexity	$\mathcal{O}(N_S)$	$\mathcal{O}(N_S)$	$\leq O(N_S)$	$\mathcal{O}(N_S)$	$\mathcal{O}(N_T)$

calculus to include memory properties within the kinematical equation.

All the presented works handle *obstacles avoidance* with basically two strategies: (i) low-level control (*EPSO*, *PPSO* and *GSO*); and (ii) artificial repulsion mechanisms (*RDPSO* and *AFS*). Despite using different algorithms within such strategies, the main idea remains the same. Low-level control strategies trigger routines whenever robots sense obstacles, thus allowing decoupling the high level behavior of robots from collision avoidance routines. Nevertheless, contrarily to the artificial repulsion mechanisms, low-level control routines do not support the integration of collision avoidance susceptibility within the algorithm behavior.

One of the common approaches in the *initial deployment* of mobile robots is using a random distribution along the scenario (*EPSO*, *PPSO*, *GSO* and *AFS*). This methodology is the simplest way of deploying robots as, in most situations, the distribution of the points of interest is usually random. However, in real situations, it is necessary to ensure several constraints of the system (e.g., *MANET* connectivity), hence increasing the complexity of the random distribution. In addition, random deployment may cause unbalanced deployment and therefore increase the hardware cost. Alternatively, the authors in [41] presented an *Extended Spiral of Theodorus* (*EST*) applied to the *RDPSO* algorithm. This methodology secures that the robots from the same swarm (i.e., cluster) are initially and autonomously deployed in an unknown environment, while avoiding areas of no interest (i.e., obstacles) and maintaining *MANET* multiple connectivity.

Most of the works consider broadcast *communication*, with purely local interactions over some specified range, in which robots only cooperate with their neighbors (*EPSO*, *PPSO* and *GSO*). Although this is the classical approach, recently many works suggested some kind of global communication without any pre-existent infrastructure, denoted as multi-hop ad hoc communication (*RDPSO* and *AFS*). This makes it possible for robots to communicate with other robots outside their direct (i.e., one-hop) range. It is noteworthy that such a strategy increases the communication overhead of the system. Nevertheless, if combined with partitioning strategies, it becomes possible to reduce the number of robots within each team, the advantages inherent to it are countless when compared to broadcast communication.

As one might expect, ensuring *MANET*s connectivity and robustness is much more demanding than infrastructured networks. As a result, to prolong the *MANET* lifetime and prevent loss of connectivity, *fault-tolerance* strategies are needed. A simple but efficient strategy is the one presented in Couceiro et al. [42], wherein robots' movements within the *RDPSO* are controlled to allow significant node redundancy guaranteeing a multi-connectivity strategy. This means that, in the worst case, a multi-connected *MANET* requires the failure of multiple robots to become disconnected. All remaining algorithms do not present any fault-tolerance strategy.

Algorithms' *parameterization* enables the calculation of values, or range of values, that would result in an improved performance. Most of the works in optimization or swarm applications present trial-and-error methodologies, not benefiting from a formal mathematical analysis. Among the previously presented algorithms, only the works of Couceiro et al. (*cf.*, [36,43]) (*RDPSO*) and Gazi and Passino [23,24] (*AFS*) present a formal analysis of their algorithms, thus restricting the parameters' definition to a range of values.

Bio-inspired algorithms usually benefit from evolutionary techniques to *avoid sup-optimal solutions*. The *RDPSO* presented by Couceiro et al. [14] handles such a problem perfectly by mimicking natural selection emulated using the principles of social exclusion and inclusion (i.e., adding and removing robots to swarms). In brief, socially active robots from the same swarm cooperate in the search task towards maximizing a given objective function (e.g., gas leak, fire outbreak, number of victims, among others). However, socially excluded robots randomly wander in the scenario instead of searching for the objective function's optimal solution like the other robots in the active swarms. However, socially excluded robots are always aware of their individual solution and the global solution of the socially excluded group. This approach improves the algorithm, making it less susceptible to becoming trapped in sub-optimum solutions. The other algorithms do not consider any specific technique to avoid sub-optimal solutions.

Similarly, only the works of Couceiro et al. (*cf.*, [14,16]) (*RDPSO*) and Krishnanand and Ghose [21,22] (*GSO*) are fitted to handle *multiple and dynamic sources*. They both use partitioning techniques for that end. Moreover, the *RDPSO* also uses an adaptive control system to systematically adjust its parameters based on contextual information [16]. This kind of adaptive mechanism is used, for instance, to control the swarm activity balancing the exploitation and exploration levels of the group or each individual agent [44,45]. The first one is related to the convergence of the algorithm, thus allowing a good short-term performance. However, if the exploitation level is too high, then the algorithm may be stuck on sub-optimal solutions. The second one is related to the diversification of the algorithm, which makes it possible to explore new solutions, thus improving the long-term performance. However, if the exploration level is too high, then the algorithm may take a long time to find the optimal solution.

The *computational complexity* refers to the system requirements for algorithm computation. The total number of robots, i.e., population, is represented by N_T. If an algorithm benefits from partitioning features or local interactions, then the number of robots

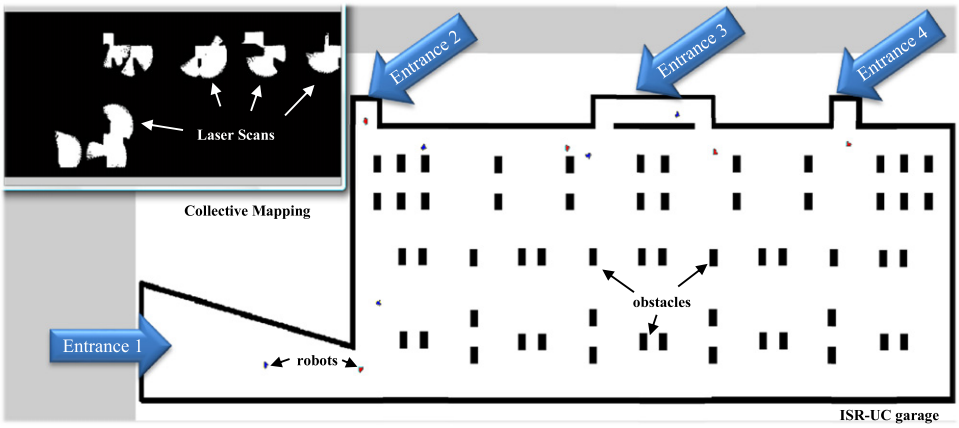

Fig. 1. *Multi-Robot Simulator* (*MRSim*). Illustration of one trial with 10 robots performing the collective mapping of an unknown scenario under the influence of the *RDPSO* algorithm. Differently colored robots represent robots from different groups/swarms [12].

within a subgroup or the broadcast signal is represented by N_S, wherein $N_S \leq N_T$. All the previously mentioned algorithms, except the *AFS*, are endowed with partitioning techniques. Nevertheless, the *RDPSO* [14,46] presents twice the computational complexity of the other algorithms that are endowed with partitioning techniques. This is due to the fault-tolerance characteristics that require the computation of a sorting algorithm (e.g., [47]).

The *memory complexity* refers to the system requirements in terms of data storage. Contrarily to the other algorithms that only require information about the previous iteration, i.e., $\mathcal{O}(1)$, the *RDPSO* exhibits a memory complexity that depends on the truncation of the fractional order series r_α (*cf.*, [37] for a more detailed description). Nevertheless, this is a difference that may be neglected since r_α is usually small and depends on the requirements of the application and the features of the robots. For instance, for the *eSwarBot* (educative Swarm Robot) platforms previously presented in [48], a $r_\alpha = 4$ leads to results of the same type as for a $r_\alpha > 4$. Although one could consider the processing power as the main reason to use a limited number of terms, the kinematical features of the platform and mission requirements also need to be considered. Hence, for *eSwarBot* platforms, the memory complexity of the *RDPSO* algorithm would be $\mathcal{O}(4)$.

The *communication complexity* refers to the local and/or global communication overhead. The algorithms that benefit from partitioning or communication broadcast strategies present a communication complexity smaller than the ones that are not endowed with such features. From the previously presented algorithms, only the *AFS* is not endowed with any of those strategies, thus resulting in a higher communication complexity, i.e., all robots within the population need to communicate with each other.

The following section presents experiments with simulated platforms so as to experimentally assess and compare the performance of the five algorithms in a search task.

3. Computational evaluation

The *Multi-Robot Simulator* (*MRSim*)[1] was used to evaluate and compare the five previously presented swarm techniques. *MRSim* is an evolution of the *Autonomous mobile robotics toolbox SIMROBOT* (*SIMulated ROBOTs*) previously developed for an obsolete version of *MatLab* [49]. The simulator was completely remodeled for the latest *MatLab* version and new features were included such as mapping and inter-robot communication. Besides, *MRSim* also makes it possible to add a monochromatic *bitmap* as a planar scenario and adjust its properties (e.g., obstacles, size, among others) by adding features of each swarm robotics technique (e.g., robotic population, maximum communication range, among others) and editing the robots' model (e.g., maximum velocity, type of sensors, among others).

This simulator was first evaluated in the context of the *CHOPIN* project [12], thus comparing decentralized and centralized versions of both *RDPSO* for exploration purposes. Fig. 1 depicts the *MRSim* interface with a simulation trial with robots using the *RDPSO* algorithm to collectively explore the whole scenario of a large basement garage environment—the *Institute of Systems and Robotics* at the *University of Coimbra* garage. This was the scenario used to compare the 5 swarm exploration algorithms previously presented as it is a large area of 2000 m^2 with a large density of obstacles (e.g., pillars).

All algorithms were evaluated while changing the number of robots within the population $N_T \in \{10, 20, 30\}$ and the maximum communication range $d_{\max} \in \{30, 100\}$ m. The communication range was based on common values presented in the literature for both *ZigBee* and *WiFi* communication (e.g., [15]). To significantly test and compare the different algorithms, 30 trials of 500 iterations for each $(N_T, d_{\max})$ combination were conducted. Also, to perform a straightforward comparison between the algorithms, robots were randomly deployed in the vicinities of each of the four entrances (see Fig. 1).

Exploring and building a map of the scenario was used as a mission objective to evaluate the five algorithms. Hence, the objective function of the team of robots was defined as a cost function in which robots need to minimize the map's entropy (*cf.*, [50] for a more detailed description), i.e., the uncertainty about map. Therefore, each robot n computes its best frontier cell as:

$$m_i^s = \underset{m_i \in \mathcal{N}(x_n[t], r_s)}{\arg\max} \left[\psi(x_t, m_i) \left\| \vec{\nabla} H(m_i) \right\| \right], \tag{1}$$

wherein $\mathcal{N}(x_n[t], r_s)$ represents the set of frontier cells located in the neighborhood of robot n with sensing radius r_s. The coefficient $\psi(x_n[t], m_i) \in [0; 1]$ measures if the cell m_i is in line-of-sight from a position $x_n[t]$, which also implies that cell m_i is likely to be empty. Moreover, the entropy of the cell m_i is represented by $H(m_i)$ and may be calculated as:

$$H(m_i) = -p(m_i) \log[p(m_i)] - (1 - p(m_i)) \log_2[1 - p(m_i)], \tag{2}$$

[1] http://www.mathworks.com/matlabcentral/fileexchange/38409-mrsim-multi-robot-simulator-v1-0.

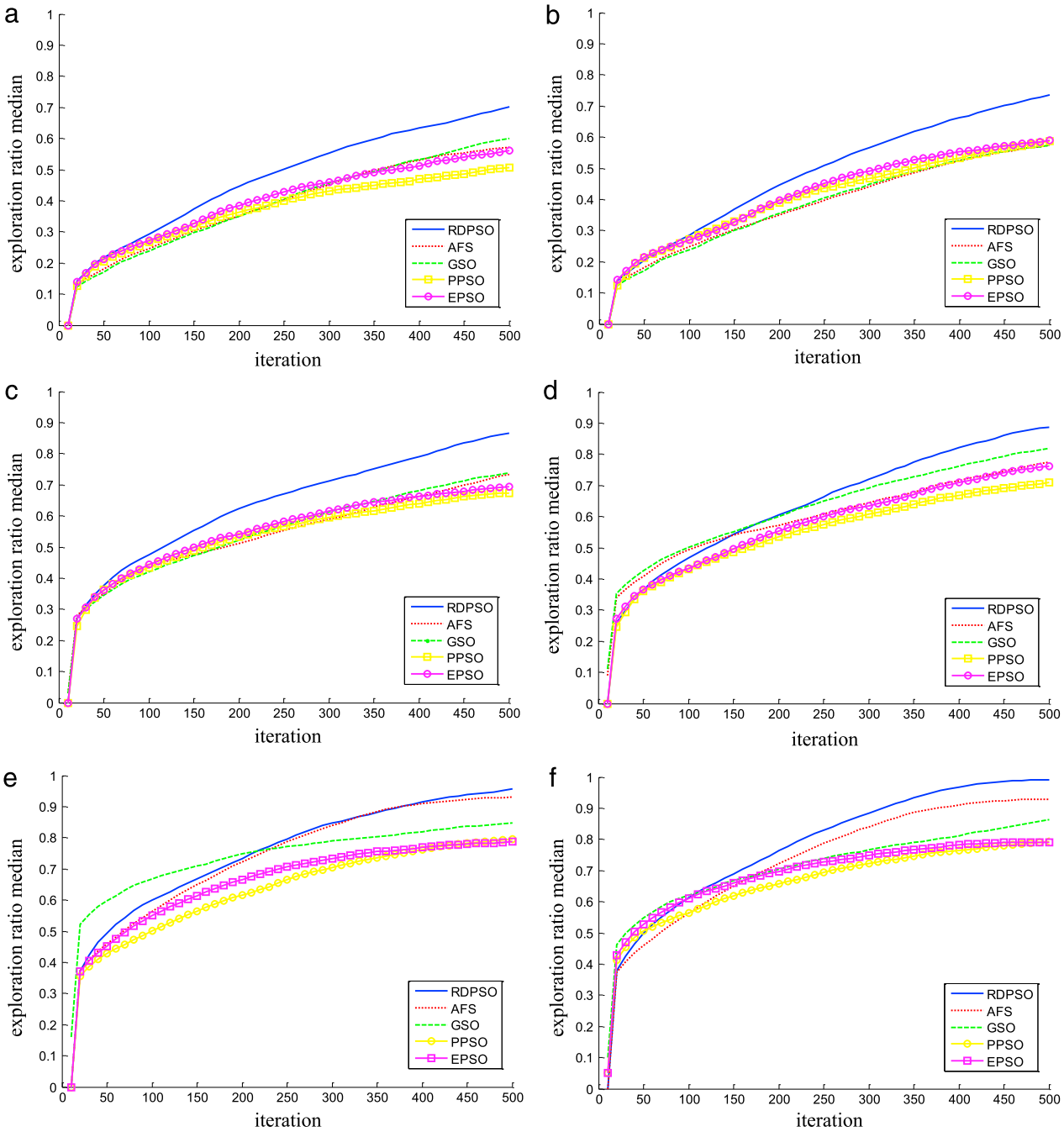

Fig. 2. Median of the exploration ratio $\eta_{\text{exp}}[t]$ over the 500 iteration for each method. (a) $(N_T, d_{\max}) = (10, 30\text{ m})$; (b) $(N_T, d_{\max}) = (10, 100\text{ m})$; (c) $(N_T, d_{\max}) = (20, 30\text{ m})$; (d) $(N_T, d_{\max}) = (20, 100\text{ m})$; (e) $(N_T, d_{\max}) = (30, 30\text{ m})$; (f) $(N_T, d_{\max}) = (30, 100\text{ m})$.

wherein $p(m_i)$ represents the probability that a grid cell is occupied.

The performance metric used is the exploration ratio of the scenario over time (number of iterations). The exploration ratio may be obtained by normalizing the mapped scenario as follows:

$$\eta_{\text{exp}}[t] = \frac{\sum A_{\text{exp}}[t]}{\sum A_{\text{real}}}, \tag{3}$$

wherein A_{real} is the matrix representing the scenario in which 0 corresponds to obstacles and 1 to free cells. Similarly, $A_{\text{exp}}[t]$ is represented by a matrix of the same size as A_{real} being the collective explored map at time, or iteration, t. Note that $\sum$ returns the sum of all matrix elements. At the beginning (step $= 0$) the collective explored map is a zero matrix, i.e., $A_{\text{exp}}[0] = \mathbf{0}$, thus resulting in an explored ratio of $\eta_{\text{exp}}[0] = 0$.

As Fig. 2 depicts, the median of the best solution over the 500 trials was taken as the final output for each $(N_T, d_{\max})$ combination.

As it is possible to observe, the *RDPSO* outperforms the other methods for all $(N_T, d_{\max})$ configurations tested. Nevertheless, such a difference decreases especially as the population of robots increases when compared to the *AFS* and the *GSO*. For instance, for the configuration of $(N_T, d_{\max}) = (30, 30\text{ m})$, i.e., Fig. 2(e), the *GSO* presents a better performance than the *RDPSO* during the first iterations while the *AFS* closely follows the same performance as the *RDPSO*.

To facilitate a straightforward comparison and since some of the algorithms present a similar performance, the area under the curve

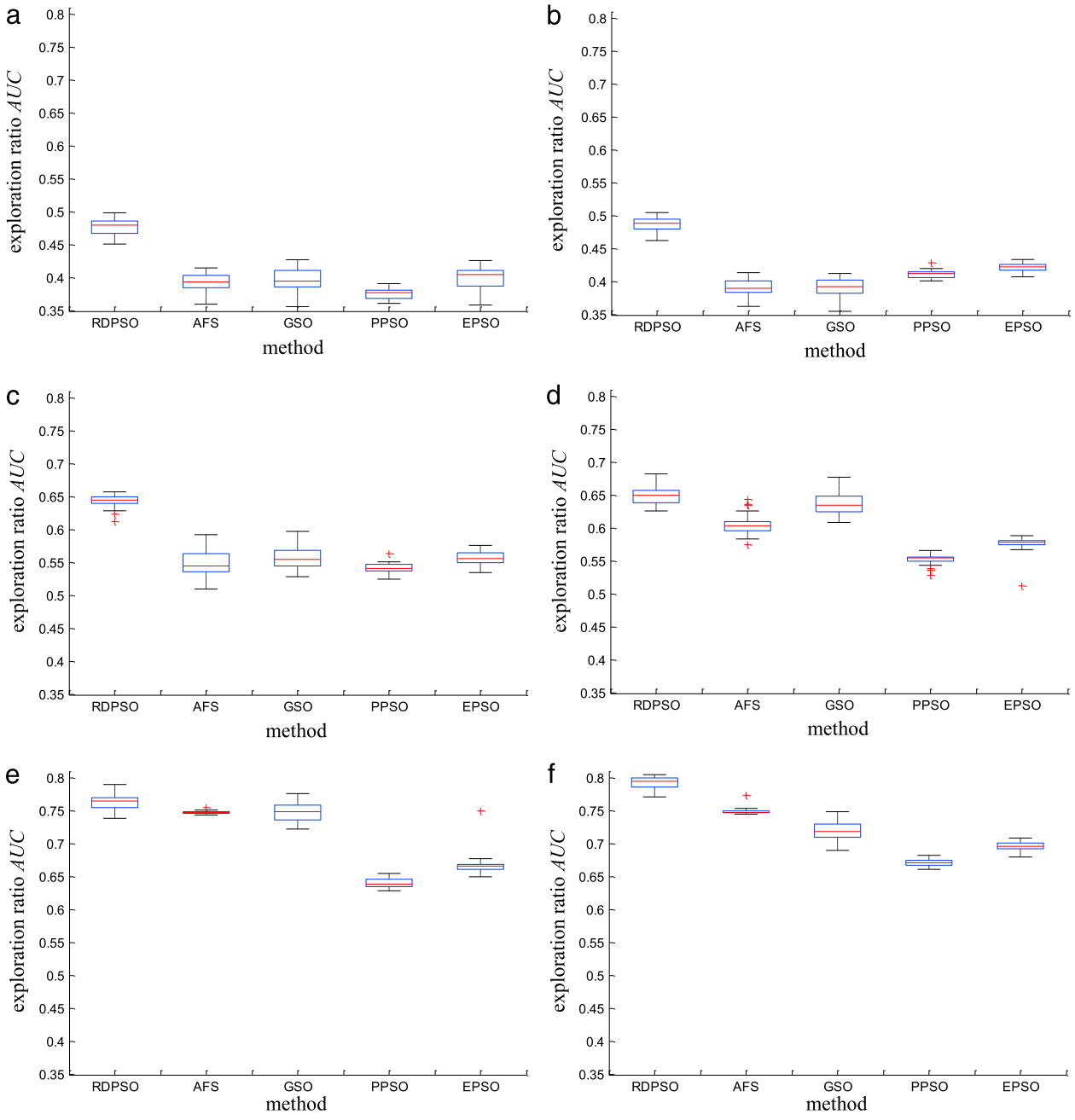

Fig. 3. *AUC* of the exploration ratio $\eta_{\text{exp}}[t]$ over the 500 iterations for each method. (a) $(N_T, d_{\max}) = (10, 30\text{ m})$; (b) $(N_T, d_{\max}) = (10, 100\text{ m})$; (c) $(N_T, d_{\max}) = (20, 30\text{ m})$; (d) $(N_T, d_{\max}) = (20, 100\text{ m})$; (e) $(N_T, d_{\max}) = (30, 30\text{ m})$; (f) $(N_T, d_{\max}) = (30, 100\text{ m})$.

(*AUC*) may be used. This is a common measure used to analyze the accuracy of receiver operating characteristic (*ROC*) curves that represent the performance of classifiers.

As the exploration ratio $\eta_{\text{exp}}[t]$ is a discrete function with $t \in \mathbb{N}_0$, the *AUC* may be calculated by the sum of each value over the 500 iterations. Moreover, one can normalize the *AUC* by dividing it by 500, thus resulting in a representation of the probability that a team of robots under a given algorithm has to explore the whole scenario. Hence, the normalized *AUC* may be calculated as:

$$AUC = \frac{1}{500}\sum_{k=0}^{500}\eta_{\text{exp}}[k]. \tag{4}$$

The *AUC* of each set of trials is represented using boxplot charts, which is a quick way of examining the algorithms' performances graphically. The ends of the blue boxes and the horizontal red line in between correspond to the first and third quartiles and the median values, respectively.

As one may observe in Fig. 3, the influence of the population is more significant than the communication range. This should be expected as swarm intelligent algorithms perform well for larger population of robots, i.e., it is possible to observe a higher degree of emergent collective behaviors as the population grows (*cf.*, [51]). Nevertheless, it is still possible to observe that, in most methods, an increase in the maximum communication range results in a minor improvement in the exploration ratio accuracy and a significant one in its precision, i.e., smaller interquartile range. In other words, the outcome becomes more predictable and regular as the maximum communication range increases. Regarding the comparison between algorithms, it is possible to observe that both

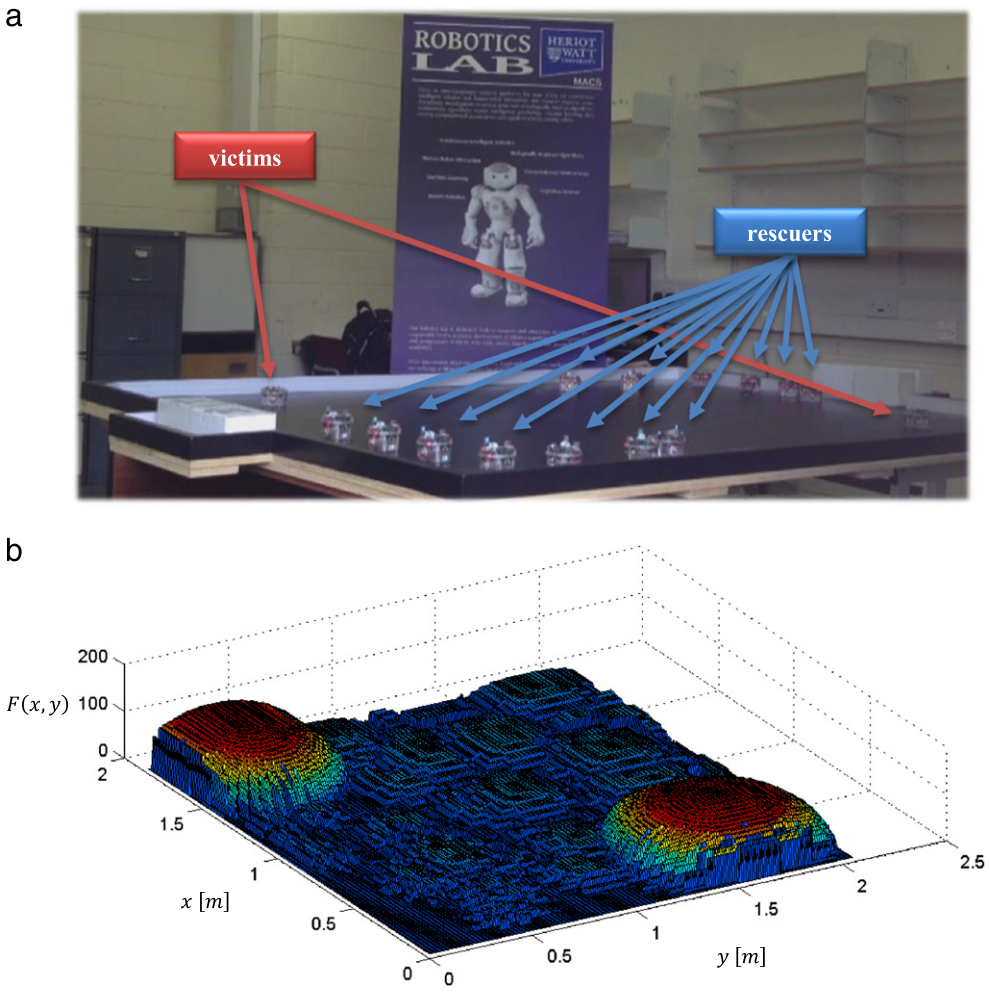

Fig. 4. Experimental setup. (a) arena of 2.0 × 1.8 m; (b) Virtual representation of the sound distribution.

PPSO and *EPSO* present a similar performance with a probability of successfully exploring the whole scenario of almost 70% for a population of 30 robots.

The same may be observed for both *AFS* and *GSO* algorithms, in which a superior performance of almost 75% may be observed for such a population. Finally, the *RDPSO* outperforms the other methods depicting a probability of successfully exploring the whole scenario of approximately 80% for the maximum population. This 5% difference may be generalized for all other $(N_T, d_{\max})$ configurations tested. Nevertheless, such a difference is not linear and although the *GSO* presents a slightly better performance than the *AFS* for smaller populations, it seems that the *AFS* is able to overcome the *GSO* as the number of robots increases. Also, and as Fig. 2 depicts, the *AFS* presents a similar performance to the *RDPSO* for larger populations of robots.

Hence, for further evaluation, the next section compares the three best performing algorithms, namely *RDPSO* [14,15], *AFS* [23,24] and *GSO* [21,22], using 14 physical robots.

4. Real experiments

In this section, the effectiveness of using the three best performing algorithms from the previous simulation experiments on swarms of *e-pucks* [52], equipped with a *Gumstix Overo COM* turret to benefit from inter-robot *WiFi* communication,[2] *b* is further explored. Due to the limitations of those turrets, all communication was centralized into a single server by means of *TCP/IP* sockets. To that end, an e-puck network manager was created on the server side to forward the data between the e-pucks and to store the necessary information to evaluate the *RDPSO*, *AFS* and *GSO* algorithms. Although this does not enable the comparison of the algorithms under different communication ranges and even paradigms (e.g., single-hop *vs* multi-hop communication), the previous experiments already considered this variable. Moreover, by not considering the *MANET* constraints, here we will only focus on evaluating the behavioral aspect of the algorithms.

With the purpose of maintaining the scope around *SaR* applications, these experiments consisted of collectively finding 2 "victims" emulated by *e-pucks* on a 2.0 × 1.8 m scenario (Fig. 4(a)). The *e-pucks* are equipped with three omnidirectional microphones that acquire data at a maximum acquisition speed of 33 kHz (A/D frequency of 100 kHz divided by three) [52]. They are also equipped with a speaker on top of them connected to an audio codec. Combined with the microphones, the speaker can create a communication network for peers' location. Unfortunately, the lack of sensitivity regarding e-pucks' microphones makes it difficult to use them for sound source localization purposes. For instance, Fig. 4(b) depicts the intensity values $F(x, y)$, with a maximum amplitude of one byte, obtained by sweeping the whole scenario with a single e-puck using the standard *SiSonic* microphones.[3] As one may see, the e-pucks are only able to

[2] http://www.gctronic.com/doc/index.php/Overo_Extension.

[3] http://projects.gctronic.com/E-Puck/docs/Audio/SP0103NC3.pdf.

distinguish sound from noise at a distance to the sound source (i.e., "victims" *e-pucks*) of approximately 30 cm. However, such a limitation favors the realistic applicability of the herein evaluated algorithms to *SaR* applications. For instance, to the similarity as the scenario used to evaluate the algorithms on simulation (Fig. 1), if one would consider a large basement garage (e.g., parking of a shopping mall), the laboratorial scenario from Fig. 4 could easily be on a scale of 1:100. As a consequence, robot rescuers would be able to "hear" victims (receiver sensitivity) at a distance of 30 m from them. Several sources would confirm that a human call for help may achieve a level between 72 and 78 dB at approximately 1 m away from the source, i.e., from a very loud voice to a shouting voice [53]. Moreover, as a rule of thumb, for every doubling of the distance from the source, the sound pressure level is reduced by 6 dB. According to [53], one may expect average ambient sound levels between 40 and 55 dB in underground structures and a medium density urban environment. This significantly reduces the ability to identify a call for help to distances between approximately 7 and 58 m, depending on the source and the ambient noise level, thus making the 30 m sensitivity of robot rescuers a realistic constraint.

The "victims" *e-pucks* were programmed to periodically play the same sound while the "rescuers" *e-pucks* were programmed with the *RDPSO*, *AFS* and *GSO* algorithms with the main objective of collectively maximizing the input retrieved by the microphones.

Contrarily to the previous experiments in which sub-optimality should be avoided to navigate towards the direction of maximum entropy at each iteration (Eq. (1)), the objective here is to find both "victims". Hence, as both *RDPSO* and *GSO* have the particularity of avoiding sub-optimality, this feature was ignored by using a simple heuristic rule to stop when retrieving a sound amplitude of 100, i.e., in the vicinities of the victims (Fig. 4(b)). This also intends to emulate the rescuing phase in which robots that found a victim should now either monitor or save it, thus being unavailable to search for other victims.

Since the 3 algorithms are stochastic, they may lead to a different trajectory convergence whenever they are executed. Therefore, test groups of 10 trials of 300 s each were considered for 14 *e-pucks*, i.e., $N_T = 14$, placed in an initial configuration as presented in Fig. 4(a).

In the case of the *RDPSO*, two swarms were initially defined dividing the whole population into two equal parts of 7 *e-pucks* each. Note that due to *RDPSO* properties, both the number of swarms and *e-pucks* within each swarm would vary during the mission based on their individual and collective performance (*cf.*, Section 2.1). In the case of both *AFS* and *GSO*, all robots belong to the same swarm. However, in the *GSO* the local-decision range varies according to the *luciferin* level, thus mimicking the same sub-division effect as the *RDPSO*.

All results from the 10 trials of each algorithm are summarized in Fig. 5. The outcome from each algorithm is represented by a different color and marker explained on the legend of the figure. Each axis corresponds to the required time to save each victim. Markers located at the borders corresponding to the 300 s depict the unsaved victims. For instance, in any of the trials the rescuers failed at finding, at least, one victim, as there is no marker on position (300, 300) s. In other words, the performance of the algorithm increases the closer to the origin (0, 0) the markers are.

As one may observe, the 3 algorithms fail at finding the 2 victims at some point over the 10 trials of 300 s in which they were each evaluated. The *RDPSO* is able to find only one victim in 2 trials, followed by the *GSO* in 4 trials and, lastly, the *AFS* in 7 trials. The outperforming of both *RDPSO* and *GSO* over the *AFS* regarding the partition of the population to multiple optimal solutions was expected due to their dynamic principles (*cf.*, Table 1). Despite not being able to always find the 2 victims,

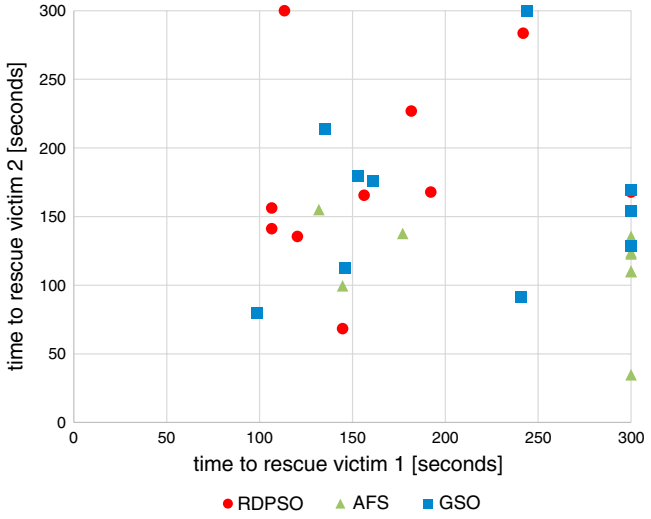

Fig. 5. Representation of the rescue success of the *RDPSO*, *AFS* and *GSO* algorithms. Each marker corresponds to a different trial under a different algorithm. The closer the markers were from the origin (0, 0), the faster the robots were able to find the victims. Markers located on the border lines of the 300 s means that only 1 victim was found during that trial.

the *AFS* presents a faster convergence, as rescuers are able to find the victim(s) in the first half of the mission time ($\lesssim$150 s). Nevertheless, this early convergence may also be the reason why rescuers may be unable to find the second victim since the *AFS* does not provide any partitioning or adaptive mechanism that could balance the already existing exploitation level of agents with higher exploration capabilities.

The performance of the *RDPSO* algorithm is closely followed by the *GSO*algorithm. The average and standard deviation times necessary to find both victims for the *RDPSO*, *AFS* and *GSO* were 205 ± 64, 259 ± 63 and 225 ± 70 s, respectively. Although both *RDPSO* and *GSO* would find all victims within a finite time due to their evolutionary mechanism to avoid stagnation, the *GSO* fails more often. As previously explained in Section 2.4, the *GSO* benefits from a *luciferin* mechanism that, contrarily to all other algorithms, does not only depend on the sensed solution (e.g., amplitude of the emitted sound by the victim). In fact, the *luciferin* value of a given robot decreases over time, thus avoiding its stagnation within a given region. We could make the analogy with nature by defining a limited quantity of oxygen in each discrete position the glowworm is in. In other words, to produce light the glowworm requires oxygen (or water) for the *enzymatic oxidation* of the *luciferin* to occur. If the glowworm has a limited amount of oxygen in a certain position (represented by the sound amplitude in these experiments), then it needs to move to another position to maintain, or even increase, its emitted light. This is a particularly interesting mechanism applied on swarm intelligence that ensures the convergence of robots to multiple solutions in an enclosed environment within a limited amount of time. However, this also plays the role of a "double-edged sword". If the robot is unable to converge quickly enough within the vicinities of a solution to maintain or increase its current *luciferin* level, then it may decide upon the wrong direction. This is likely to happen under noisy and nonlinear measures such as sound propagation with an increased complexity added by the lack of sensitivity of e-pucks' microphones. This phenomenon was observed on some occasions during the experiments in which clusters of robots within the *GSO* got close enough to listen to the victim but still depicted a poor convergence when converging at all. It is noteworthy that this could possibly be overcome by tuning parameters ρ and γ

from Algorithm 4 though little insights are introduced in [21,22] regarding those.

A video of the experiments is provided to better understand the typical behavior of the 3 algorithms under these experiments.[4] Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.robot.2013.10.004.

5. Discussion

The authors would like to discuss the take-home message this paper brings forth.

The primary motivation for this work was to find a group of swarm robotics algorithms with the potential of fulfilling realistic search tasks such as *SaR* operations. From that initial theoretical survey, five algorithms were chosen, namely: the *RDPSO* (recently proposed by the authors) [14,15], the *EPSO* [17,18], the *PPSO* [19,20], the *GSO* [21,22] and the *AFS* [23,24]. Table 1 was the leading step towards a detailed comparison of the five algorithms, thus describing the most relevant features one should expect under such tasks. From that table it was possible to conclude that the *RDPSO* touches upon all desired features for a higher computational and memory cost. The main features separating the *RDPSO* from the alternatives were its ability to avoid sub-optimality by benefitting from a punish–reward mechanism based on natural selection [14] and the fault-tolerance assessment using a multi-connectivity strategy [42]. Such an outcome promotes the use of the *RDPSO* algorithm in applications affected by multiple sub-optimal and dynamic solutions in which the communication may be susceptible to failures. However, both CPU power and the memory of the robotic platforms need to be well-weighted due to the requirements of the *RDPSO*.

Going deeper into the "rabbit hole", a large number of simulation experiments were conducted to study the effect of the number of robots and the communication constraints of the five algorithms. The mission consisted of exploring and mapping a 2000 m^2 scenario in which robots needed to minimize the map's entropy [50]. More than to just state the obvious phenomenon that a larger population of robots improves the overall performance, those experiments were useful to understand the influence of a more constrained communication network on the five swarm algorithms. Through Fig. 3 it was possible to observe a lower variability of the exploration ratio for a larger maximum communication distance feasible between robots, i.e., the outcome became more consistent for a less constrained communication network. Such a phenomenon was more perceptible using the *EPSO* and *PPSO* algorithms, thus suggesting their higher susceptibility over the communication constraints. Associating this aspect to the fact that both algorithms work on a broadcast communication basis (*cf.*, Table 1), the authors dissuade the use of those algorithms on applications that may require a larger number of robots (above 20 in the experiments in Section 3) or too limited communication constraints (below an inter-robot distance of 100 m in the experiments in Section 3).

Those results paved the way to an insightful evaluation of the three best performing algorithms, namely, the *RDPSO*, the *GSO* and the *AFS*. This new evaluation was carried out using real platforms: the well-known *e-puck* robots equipped with *WiFi* technology for inter-robot communication [52]. Instead of a mapping mission that would be typical of a *reconnaissance* phase (*cf.*, Section 2 and Couceiro et al. work [12]), those experiments were consistent with the next phase of the firefighting operation, the *rescuing*. In brief, these experiments consisted of collectively finding 2 "victims" by benefiting from *e-pucks*' speakers and microphones. To complement the previous experiments in which the size of the population and the communication constraints were studied, these experiments were conducted to evaluate the behavioral aspect, and even evolutionary features, of the algorithms. The results fostered even more the use of the *RDPSO* for such tasks with a 80% success of finding both victims over the 300 s. Nevertheless, the *GSO* was able to closely follow the *RDPSO* due to its evolutionary *luciferin* mechanism for stagnation avoidance. Such a result proves to be crucial since the *GSO* presents itself as a "low cost" alternative to the *RDPSO* in terms of computational and memory requirements. Although, in general, the *RDPSO* presented better results than the *GSO*, it is noteworthy that the *GSO* would achieve a similar final outcome if one could benefit from a larger mission time.

All that being said, one may state that it is still difficult at this point to find a simple answer to the question "which is the best swarm robotics algorithm for my application?". However, the authors argue that this paper provides a preliminary rationale on the most fitted swarm robotics algorithm for search applications. Such a choice should consider some predefined assumptions, such as the number of available robots, the existing wireless communication and other mission-related features (e.g., existence of dynamic sources, number of sub-optimal solutions, among others).

6. Conclusion and future work

One of the main questions regarding swarm robotics algorithms is whether the full-scale deployment of these systems in real-world application environments would fit the necessary mission requirements. Despite the outstanding accomplishment of such algorithms in optimization or any other task unconstrained by real world features, such as robot dynamics, obstacles interference or communication failures, the reality gap still needs to be crossed for most of them. To address this issue, this paper outlined an initial benchmark regarding the outcome from five swarm robotics algorithms under different configurations (e.g., number of robots) and search tasks. Such results can be used to apply swarm robotics concepts to real world applications such as search-and-rescue.

The list of swarm robotics algorithms compared in this paper is by no means exhaustive and a deeper research should be conducted based on the insights provided in this paper. It is, however, possible to make a proper selection of the most desired algorithm based on the requirements of the application and hardware limitations (e.g., wireless technology).

The experimental results essentially show the advantages of using evolutionary algorithms over non-evolutionary ones, starting with simulation experiments in which robots need to cooperatively map an unknown environment, and all the way to real experiments in which a group of e-pucks needs to find the location of victims through sound. With a small increase in the computational complexity, the Robotic Darwinian Particle Swarm Optimization (*RDPSO*) algorithm depicts an improved convergence which is also better fitted to handling multiple and dynamic sources.

Given the advantages of the *RDPSO* algorithm, a deeper analysis and comparison should be conducted as a future work. Moreover, a macroscopic model of the *RDPSO* should be developed in order to predict teams' performance for a given task. By doing this, one may be able to choose the most correct configuration (e.g., number of robots within each team) without resorting to exhaustive experimentation.

[4] http://www2.isr.uc.pt/~micaelcouceiro/media/RDPSO_AFS_GSO.mp4.

Acknowledgments

This work was supported by a *Ph.D.* scholarship (SFRH/BD /73382/2010) granted to the first author and the research project *CHOPIN* (PTDC/EEA-CRO/119000/2010), both funded by the Portuguese Foundation for Science and Technology (*FCT*). This work was made possible by the School of Mathematical and Computer Sciences from Heriot-Watt University that provided the necessary resources to conduct the real experiments. The authors would also like to thank Monica Ivanova for her cooperation and advice.

References

[1] S.J. Benkoski, M.G. Monticino, J.R. Weisinger, A survey of the search theory literature, Naval Res. Logist. 31 (4) (1991) 469–494.
[2] J. Suarez, R. Murphy, A survey of animal foraging for directed, persistent search by rescue robotics, in: Proceedings of the 2011 IEEE International Symposium on Safety, Security and Rescue Robotics, Kyoto, Japan, 2011.
[3] R.R. Murphy, Human–robot interaction in rescue robotics, IEEE Trans. Syst. Man Cybern. 34 (2) (2004) 138–153.
[4] D. Floreano, C. Mattiussi, Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies, MIT Press, Cambridge, MA, 2008.
[5] M. Dorigo, E. Sahin, Swarm robotics—special issue, Auton. Robots 17 (2004) 111–113.
[6] S. Kernbach, D. Häbe, O. Kernbach, R. Thenius, G. Radspieler, T. Kimura, T. Schmickl, Adaptive collective decision-making in limited robot swarms without communication, Int. J. Robot. Res. 32 (1) (2013) 35–55.
[7] M. Li, A. Alvarez, F. De Pellegrini, B. Prabhakaran, I. Chlamtac, ROBOTRAK: a centralized real-time monitoring, control, and coordination system for robot swarms, in: Proceedings of the 1st International Conference on Robot Communication and Coordination, RoboComm'07, IEEE Press, Athens, Greece, 2007.
[8] E. Sahin, Swarm robotics: from sources of inspiration to domains of application, Lecture Notes in Comput. Sci. 3342 (2005) (2005) 10–20.
[9] L.E. Parker, Distributed intelligence: overview of the field and its application in multi-robot systems, J. Phys. Agents 2 (1) (2008) 5–14.
[10] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, New York, 1999.
[11] R.P. Rocha, 2012. [Online] Available: http://chopin.isr.uc.pt/.
[12] M.S. Couceiro, D. Portugal, R.P. Rocha, A collective robotic architecture in search and rescue scenarios, in: Proceedings of the 28th Symposium on Applied Computing, SAC2013, Coimbra, Portugal, 2013.
[13] J. Kennedy, R. Eberhart, A new optimizer using particle swarm theory, in: Proceedings of the IEEE Sixth International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.
[14] M.S. Couceiro, R.P. Rocha, N.M.F. Ferreira, A novel multi-robot exploration approach based on particle swarm optimization algorithms, in: IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR2011, Kyoto, Japan, 2011.
[15] M.S. Couceiro, R.P. Rocha, N.M.F. Ferreira, Ensuring ad hoc connectivity in distributed search with robotic darwinian swarms, in: Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR2011, Kyoto, Japan, 2011.
[16] M.S. Couceiro, J.A.T. Machado, R.P. Rocha, N.M.F. Ferreira, A fuzzified systematic adjustment of the robotic darwinian PSO, Robot. Auton. Syst. (2012).
[17] J. Pugh, A. Martinoli, Multi-robot learning with particle swarm optimization, in: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, 2006.
[18] J. Pugh, A. Martinoli, Inspiring and modeling multi-robot search with particle swarm optimization, in: Proceedings of the 2007 IEEE Swarm Intelligence Symposium, 2007.
[19] J. Hereford, M. Siebold, Multi-robot search using a physically-embedded particle swarm optimization, Int. J. Comput. Intell. Res. 4 (2) (2008) 197–209.
[20] J.M. Hereford, M.A. Siebold, Bio-inspired search strategies for robot swarms, in: Swarm Robotics, From Biology to Robotics, 2010.
[21] K.N. Krishnanand, D. Ghose, A glowworm swarm optimization based multi-robot system for signal source localization, in: Design and Control of Intelligent Robotic Systems, in: Studies in Computational Intelligence, 2009, pp. 49–68.
[22] K.N. Krishnanand, D. Ghose, Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions, Swarm Intell. 3 (2009) 87–124.
[23] V. Gazi, K.M. Passino, Stability analysis of swarms, IEEE Trans. Automat. Control 48 (4) (2003) 692–697.
[24] V. Gazi, K.M. Passino, Stability analysis of social foraging swarms, IEEE Trans. Syst. Man Cybern. 34 (1) (2004) 539–557.
[25] Y.d. Valle, G.K. Venayagamoorthy, S. Mohagheghi, J.C. Hernandez, R. Harley, Particle swarm optimization: basic concepts, variants and applications in power systems, IEEE Trans. Evol. Comput. 2 (2) (2008) 171–195.
[26] J. Tang, J. Zhu, Z. Sun, A novel path planning approach based on appart and particle swarm optimization, in: Proceedings of the 2nd International Symposium on Neural Networks, in: LNCS, vol. 3498, 2005.
[27] E.J.S. Pires, P.B.d.M. Oliveira, J.A.T. Machado, J.B. Cunha, Particle swarm optimization versus genetic algorithm in manipulator trajectory planning, in: 7th Portuguese Conference on Automatic Control, Lisbon, Portugal, 2006.
[28] M.S. Couceiro, R. Mendes, N.M. Fonseca Ferreira, J.A. Tenreiro Machado, Control optimization of a robotic bird, in: European Workshop on Movement Science, EWOMS'09, Lisbon, Portugal, 2009.
[29] M.S. Couceiro, J.M.A. Luz, C.M. Figueiredo, N.M.F. Ferreira, Modeling and control of biologically inspired flying robots, Robotica J. Camb. 30 (1) (2012) 107–121.
[30] P. Ghamisi, M.S. Couceiro, J.A. Benediktsson, N.M.F. Ferreira, An efficient method for segmentation of images based on fractional calculus and natural selection, Expert Syst. Appl. 39 (16) (2012) 12407–12417. Elsevier.
[31] M.R. Alrashidi, M.E. El-Hawary, A survey of particle swarm optimization applications in power system operations, Electr. Power Compon. Syst. 34 (12) (2006) 1349–1357.
[32] M.S. Couceiro, J.M.A. Luz, C.M. Figueiredo, N.M.F. Ferreira, G. Dias, Parameter estimation for a mathematical model of the golf putting, in: Proceedings of WACI-Workshop Applications of Computational Intelligence, Coimbra, Portugal, 2010.
[33] L. Marques, A.T.d. Almeida, Finding odours across large search spaces: a particle swarm-based approach, in: Proc. 6th Intl. Conf. on Climbing & Walking Robots, CLAWAR, Madrid, Spain, 2004.
[34] W. Jatmiko, K. Sekiyama, T. Fukuda, Modified particle swarm robotic odor source localization in dynamic environments, Int. J. Intell. Control Syst. 11 (2) (2006) 176–184.
[35] M.S. Couceiro, F.M.L. Martins, F. Clemente, R.P. Rocha, N.M.F. Ferreira, Towards a further understanding of the robotic darwinian PSO, in: Computational Intelligence and Decision Making—Trends and Applications, in: From Intelligent Systems, Control and Automation: Science and Engineering Bookseries, Springer-Verlag, 2012, pp. 17–26.
[36] M.S. Couceiro, F.M.L. Martins, R.P. Rocha, N.M.F. Ferreira, Analysis and Parameter Adjustment of the RDPSO—Towards an Understanding of Robotic Network Dynamic Partitioning based on Darwin's Theory, Vol. 7, International Mathematical Forum, Hikari, Ltd., 2012, pp. 1587–1601. No. 32.
[37] M.S. Couceiro, F.M.L. Martins, R.P. Rocha, N.M.F. Ferreira, Introducing the fractional order robotic darwinian PSO, in: Proceedings of the 9th International Conference on Mathematical Problems in Engineering, Aerospace and Sciences—ICNPAA'2012, Vienna, Austria, 2012.
[38] V. Braitenberg, Vehicles, in: Experiments in Synthetic Psychology, The MIT Press, 1984.
[39] O. Michel, Webots: professional mobile robot simulation, J. Adv. Robot. Syst. 1 (2004) 39–42.
[40] K. Warburton, J. Lazarus, Tendency-distance models of social cohesion in animal groups, J. Theoret. Biol. 150 (1991) 473–488.
[41] M.S. Couceiro, C.M. Figueiredo, R.P. Rocha, N.M.F. Ferreira, Darwinian swarm exploration under communication constraints: initial deployment and fault-tolerance assessment, Robot. Auton. Syst. (2013) submitted for publication.
[42] M.S. Couceiro, R.P. Rocha, N.M.F. Ferreira, Fault-tolerance assessment of a darwinian swarm exploration algorithm under communication constraints, in: Proc. of 2013 IEEE International Conference on Robotics and Automation, ICRA 2013, Karlsruhe, Germany, 2013.
[43] M.S. Couceiro, F.M.L. Martins, R.P. Rocha, N.M.F. Ferreira, Mechanism and convergence analysis of a multi-robot swarm, J. Intell. Robot. Syst. (2013) submitted for publication.
[44] K. Yasuda, N. Iwasaki, G. Ueno, E. Aiyoshi, Particle swarm optimization: a numerical stability analysis and parameter adjustment based on swarm activity, IEEJ Trans. Electr. Electron. Eng. 3 (2008) 642–659. Wiley InterScience.
[45] Y. Wakasa, K. Tanaka, Y. Nishimura, Control-theoretic analysis of exploitation and exploration of the PSO algorithm, in: IEEE International Symposium on Computer-Aided Control System Design, IEEE Multi-Conference on Systems and Control, Yokohama, Japan, 2010.
[46] M.S. Couceiro, R.P. Rocha, N.M.F. Ferreira, P.A. Vargas, Darwinian robotic swarms for exploration with minimal communication, in: IEEE Congress on Evolutionary Computation, CEC'2013, Special session on Evolutionary Robotics, 2013, Cancún, México, 2013.
[47] P. Bhalchandra, N. Deshmukh, S. Lokhande, S. Phulari, A comprehensive note on complexity issues in sorting algorithms, Adv. Comput. Res. 1 (2) (2009) 1–9.
[48] M.S. Couceiro, C.M. Figueiredo, J.M.A. Luz, N.M.F. Ferreira, R.P. Rocha, A low-cost educational platform for swarm robotics, Int. J. Robots Educ. Art (2011).
[49] University of Technology, 2001. [Online] Available: http://www.uamt.feec.vutbr.cz/robotics/simulations/amrt/simrobot_en.html.
[50] R.P. Rocha, F. Ferreira, J. Dias, Multi-robot complete exploration using hill climbing and topological recovery, in: Proc. of 2008 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'2008, Nice, France, 2008.
[51] G. Beni, From swarm intelligence to swarm robotics, in: Proceedings of the Swarm Robotics Workshop, Heidelberg, Germany, 2004.
[52] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.C. Zufferey, D. Floreano, A. Martinoli, The e-puck—a robot designed for education in engineering, in: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions.
[53] B. Truax, Handbook for Acoustic Ecology, second ed., World Soundscape Project, Simon Fraser University and ARC Publications, 1999.

Micael S. Couceiro is a Doctoral Candidate in Electrical and Computer Engineering at the Faculty of Sciences and Technology of University of Coimbra. He obtained the M.Sc. degree in Automation and Communications in Energy Systems — Industrial Systems in April 2010, at the Engineering Institute of Coimbra — Portugal.

He conducts research on multi-robot systems and swarm robotics at the Mobile Robotics Laboratory, a part of the Institute of Systems and Robotics, in Coimbra, Portugal. He has published papers on Mobile Robotics, Biomimetics, Fractional-Order Control, Sports Engineering, Biomechanics and Mathematical Methods.

Patricia A. Vargas received her Ph.D. in Computer Engineering from the University of Campinas, Unicamp (Brazil) in 2005. She is currently Director of the Robotics Laboratory and Lecturer in Computer Science and Robotics at the School of Mathematical and Computer Science at Heriot-Watt University (Edinburgh, Scotland, UK). She was a post-doc at the Centre for Computational Neuroscience and Robotics, University of Sussex (England, UK) for 3 years. Her research interests include Evolutionary and Bio-inspired Robotics, Swarm Robotics, Computational Neuroscience, Data Mining and Machine Learning, Biologically-inspired algorithms and Human–Robot Interaction.

Rui P. Rocha completed his Ph.D. degree on May 2006, at the Faculty of Engineering of the University of Porto. Between February 2000 and May 2006, he was a Teaching Assistant at the Department of Electrical and Computer Engineering, in the Faculty of Sciences and Technology of the University of Coimbra.

Currently he is an Assistant Professor at the Department of Electrical and Computer Engineering and a Researcher at the Institute of Systems and Robotics, in the Faculty of Sciences and Technology University of Coimbra. His main research topics are cooperative Multi-Robot Systems, 3D Map Building, Distributed Architectures and Intelligent Transportation Systems.

Nuno M.F. Ferreira graduated with a B.Sc. degree from the Engineering Institute of Coimbra in Electrical Engineering (1994). He graduated with a Teaching Licensure (B.Sc. plus 2 years) and MS.c. at the University of Porto in Electrical Engineering (1999). He obtained the Ph.D. degree in Electrical Engineering at the Faculty of Engineering of the University of Trás-os-Montes e Alto Douro (2006).

Since 1997 he has worked at the Engineering Institute of Coimbra and he is currently a Professor in the Department of Electrical Engineering. His main research topics are Robotics and Control Systems.