Improving Search Capabilities of Reduced Real Swarm of Robots by Virtual Particle Swarm

Otávio José dos Santos
University of Pernambuco
Recife, Pernambuco 50720-001
Email: s.otavio@gmail.com

Sergio Campello Oliveira
University of Pernambuco
Recife, Pernambuco 50720-001
Email: sergio.campello@upe.br

***Abstract*—This paper introduces a new approach to improve the search capabilities of a reduced real swarm of robots based on the classical Particle Swarm Optimization (PSO) by employing a larger virtual swarm. The central idea is to add more particles in the virtual swarm using them to aid the real search. This approach aims to reduce costs once that increasing the quantity of robots in the real swarm implies higher costs related to robots acquisition and its maintenance. To work around this issue, we insert particles only in the virtual swarm. These particles are able to interact with the real robots and help them. The description functions of virtual and real environment are slightly different to represent dynamic changes on real data environment. Preliminary results indicates that the virtual particles aid the real ones improving its search mechanisms. The virtual aided the few number of real robots obtained equivalent solutions than the larger number of robots in the same search space.**

***Keywords*—Swarm Intelligence; Particle Swarm Optimization; Swarm of Robots**

I. Introduction

There are situations that a certain environment can be affected and changed as such in disaster contexts. In these cases, the changes in the environment are distributed in specific points and with a limited coverage. That is, the altered environment kept some characteristics that original one. Thus, if the original environment is previously known, it is possible to use the original informations for navigation in the environment.

Employ robots to hazardous tasks is an alternative to safeguard human lives. In this context, the use of robots in disaster environments is the focus of various research groups [1]. One of the strategies is to use a group of robots to search the environment. The target of the search vary depending on the objective determined for the robots group.

When these robots act in a community way, cooperating to achieve a common objective, they are referenced as a swarm of robot. In this swarm, the robots share the information obtained individually about the environment to help the swarm perform certain task. From the interaction among the individuals capable of adapting itself to changes in the environment through feedback mechanism, emerges the collective behavior [2]. In this context, arises the swarm intelligence concept.

The swarm intelligence can be defined as a collective behavior of self-organized particles with a decentralized controller that is, usually, observed in the nature [3].

The goal in optimization problems is achieve the minimum (maximum) value of the objective function. The objective function is the function that describe the system behavior. In general, such function is not known hence the optimization techniques try to find its minimum (maximum) value through an estimate or approximation.

If a problem can be transformed into an optimization problem, it can be associated to swarm intelligence in order to obtain its solution [4]. Thus, tasks like path definition and target search can be handle by swarm intelligence algorithms.

Swarm of robots are think to be robustness, scalables and flexibles [5]. In swarm paradigm, robustness is the ability to handle members losses without damage the swarm; scalability refers to property of keep swarm performance with the introduction or removal of member in the group; flexibility is the ability to handles different tasks and environments [5] [6].

In real-life application, usually, the employ of swarms of robots happens through small swarms [7]. This is due because the costs associate with robots. The costs to acquire a group of robots and to do periodic maintenance are some of the limitations. Moreover, depending on the kind of robot, the infrastructure to operate a large swarm can be costly; for example with communication data transfer. Thereby, improve the capabilities of small swarm of robots can aid to reduce the outgoings and work around the involved issues without apply a large swarm of robots.

II. Particle Swarm Optimization

Among the bio-inspired algorithms, exist a subgroup of algorithms which takes inspiration from social interaction of animals swarms such as bird flocks, ant colony and fish school.

The Particle Swarm Optimization (PSO) is a bio-inspired algorithm for optimization problems introduced by Kennedy and Eberhart [8], in 1995. Its computational implementation is simple and is grounded on determining parameters that change the motion trend (velocity) of each member of the swarm. For it, the individuals of the swarm search the environment and make a movement through the evaluation of the information obtained by itself and its neighbors [8].

This algorithm is widely known and used in different applications scenarios.

In the PSO algorithm each member of the swarm is known as particle. Each particle is a candidate solution for the problem. Each particle changes its position according to its own experience and the experience of the other members of the swarm [9].

To assess the performance of the solution, it is necessary to define a fitness function which is responsible for indicating how good is the solution achieved by each particle [9]. This function is an essential part of the algorithm since it influences directly the movement process of the particles [9].

Furthermore, Shi and Eberhart [10] in 1998 introduced the concept of inertia weight that affects the velocity of the particle. This parameter is responsible to refine the exploration and exploitation control [11].

Regarding a swarm of particles with P particles, in a given instant t, each particle p in the D-dimensional search space is represented by a position $\mathbf{x}_p = \{x_{p_1}, x_{p_2}, x_{p_3}, ..., x_{p_D}\}$ and by a velocity $\mathbf{v}_p = \{v_{p_1}, v_{p_2}, v_{p_3}, ..., v_{p_D}\}$.

By a iterative method, the particle movement is given by Equation 1 and the velocity v_p by Equation 2, where t represents the current iteration.

$$\mathbf{x}_p^{t+1} = \mathbf{x}_p^t + \mathbf{v}_p^t \tag{1}$$

$$\mathbf{v}_p^{t+1} = w_p \mathbf{v}_p^t + c_{p_1} \mathbf{r}_{p_1} * (\mathbf{pbest}_p - \mathbf{x}_p^t) + c_{p_2} \mathbf{r}_{p_2} * (\mathbf{gbest} - \mathbf{x}_p^t) \tag{2}$$

in which w denotes a inertia weight; $\mathbf{r}_{p_1} = \{r_{p_1}, r_{p_2}, ..., r_{p_D}\}$ and $\mathbf{r}_{p_2} = \{r_{p_1}, r_{p_2}, ..., r_{p_D}\}$ are random vectors generated by a uniform distribution in the interval [0,1]; c_{p_1} and c_{p_2} are, respectively, cognitive and social constants/rates; $\mathbf{pbest}_p$ is the best position found by particle p until the iteration t, that is, the position $\mathbf{x}_p$ in which the fitness function of particle p returned the best result; **gbest** is the best position found by the swarm or by a set of neighbors of p.

In order to restrict the velocity of the particle in each dimension, a parameter $\mathbf{v}_{p_{max}} = [-v_{max}, v_{max}]$ is applied. In the execution of the algorithm, when a velocity $\mathbf{v}_p$ is evaluated and this velocity is out of the range $[-v_{max}, v_{max}]$ of $\mathbf{v}_{p_{max}}$, $\mathbf{v}_p$ is set to respective limit defined in $\mathbf{v}_{p_{max}}$.

A parameter $\mathbf{b}_p = \{b_{p_1}, b_{p_2}, b_{p_3}, ..., b_{p_D}\}$ is responsible to restrict the bounds of the search space where the swarm acts. If the position $\mathbf{x}_p$ of a particle is evaluated and it is out of the bounds of the search space, $\mathbf{x}_p$ is set to respective limit defined in $\mathbf{b}_p$.

All particles are considered similar to each other, thus, as default, the parameters c_{p_1}, c_{p_2}, w_p, $\mathbf{v}_{max}$, $\mathbf{b}_p$ are equals for all particles.

For minimization problem, we wish for find $\mathbf{x}_p$ minimizes the function. The classical PSO Algorithm is shown in Algorithm 1.

Algorithm 1: PSO Pseudo-code

```
1  Initialize swarm/particles parameters;
2  Initialize particles in the search space;
3  Be $f(\mathbf{x}_p)$ the fitness function of particle $p$;
4  gbest ← $\mathbf{x}_0$;
5  while stop condition not satisfied do
6      foreach particle $p$ do
7          Generate the random vectors $\mathbf{r}_{p_1}$ and $\mathbf{r}_{p_2}$;
8          Update velocity $\mathbf{v}_p$;
9          Update position $\mathbf{x}_p$;
10         Evaluate fitness $f(\mathbf{x}_p)$;
11         Update $\mathbf{pbest}_p$ and gbest ;
12     end
13 end
14 return $f(\mathbf{gbest})$
```

III. Proposed Technique/Method

Considering a D-dimensional space S described through a vector $\mathbf{x} = \{x_1, x_2, x_3, ..., x_D\}$. We assume that some properties of the space can be mapped to a function f. For robots application, this space is associated with the environment where they operate, thus $f(\mathbf{x})$ represents a map of original environment.

If a change occurs in the environment, the original map is degraded and a new function $h(\mathbf{x})$ is a better candidate to map the altered environment. Depending on level of changes in the environment, $f(\mathbf{x})$ function can still keep representative characteristics of the altered/new environment. With this assumption, h can be interpreted as a combination of f and another function g which the last one model the degradation in the environment. If the level of degradation in the environment is low, f tend to h as we can observe in Equation 3.

$$h(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}) \tag{3}$$

To better understand, let f be the three dimensional Sphere function which represents the original environment. The original environment is shown in Figure 1(a). In a given moment occurs a change in that environment. This change will be represented by the three dimensional Rastrigin function g. Thus, h function which represents the degraded environment is given by a combination of Sphere and Rastrigin function shown in Figure 1(b).

In the same way, a original environment represented by the three dimensional Rastrigin function shown in Figure 1(c). The changes in this environment are represented by the three dimensional Ackley function. The new environment, that is, after degradation is shown in Figure 1(d).

In both examples, the changes are applied centered in the origin and limited in range [-1,1] in each dimension.

The performance of a small swarm of robots operating in a environment could be increased with the addition of more robots in this swarm. However, increase in this number implicates higher costs with hardware and infrastructure to control them. To balance this deadlock, we can increase the number of members in the swarm employing virtual particles able to assist the robots in the real environment.

While the robots operate in the real world, virtual particles operate in a simulated computational world which resembles the real one. These two scenarios can be associated to original environment represented by f and degraded environment represented by h respectively.

In a real context that functions are defined according to application and they depend on environment and robots char-

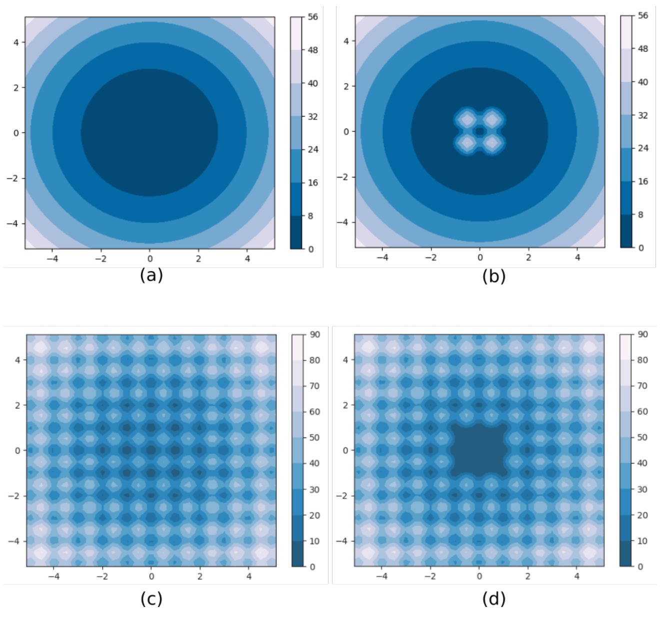

Fig. 1. Three Dimensional Environments: original environments (a) and (c) and correspondent degraded environments (b) and (d).

acteristics. Some important variables in robot context are, for example, battery level and communication signal level and they can be consider in the fitness calculation. For example, under some conditions, a low battery level can penalize the evaluated fitness to save energy and ensure the robot achieve the end of the mission. Another example: a robot with good signal level may have the evaluated fitness improved because it owns better chances to relaying a message than other robots. In both examples, the functions f and h can be affected by different ways.

In this context, we can create a hybrid swarm composed by the virtual particles and the robots. As in a common swarm, the members of hybrid swarm are able to exchange information and cooperate with each other to achieve a mutual objective. However, they operate in different search spaces.

In general, these virtual particles can mirror the characteristics and behavior of the real robots or simply a set of that characteristics and behavior in view of to save computational resources.

Therefore, an adaptation in the PSO Algorithm is necessary to cover the situation cited. First, we split the original swarm in two set of particles, one corresponding to virtual particles and other to real robots. The second one is used to store the information obtained for robots in the real world. Both acts as a single swarm, sharing same **gbest**, however each particles set has its own fitness function.

Along this paper, we will use the index p for virtual particles in the set of size P and index q for particles bounded/associated to robots; Q denotes the number of real robots in the swarm.

To associate the particles with robots we make a binding process: for each robot in the swarm, its state is copied to a virtual particle in order to this particle reproduce the real-life robot behavior into computational simulation. The state of the robot is represented by a D-dimensional vector which stores significant informations of the robot. In PSO approach, this vector is the position vector $\mathbf{x}$.

The binding process is described in Algorithm 2.

Algorithm 2: BindProcess() Pseudo-code (input: number of robots Q)

```
1 for robot $q \leftarrow 0$ to $Q$ do
2     Copy state of the robot $q$ to particle $\mathbf{x}_q$;
3 end
```

To model the movement of virtualized robots, Equation 1 and Equation 2 can be rewritten as Equation 4 and Equation 5.

$$\mathbf{x}_q^{t+1} = \mathbf{x}_q^t + \mathbf{v}_q^t \tag{4}$$

$$\mathbf{v}_q^{t+1} = w_q \mathbf{v}_q^t + c_{q_1} \mathbf{r}_{q_1} * (\mathbf{pbest}_q - \mathbf{x}_q^t) + c_{q_2} \mathbf{r}_{q_2} * (\mathbf{gbest} - \mathbf{x}_q^t) \tag{5}$$

where w_q, c_{q_1}, c_{q_2}, r_{q_1}, r_{q_2}, $pbest_q$ are, respectively, the equivalents to w_p, c_{p_1}, c_{p_2}, r_{p_1}, r_{p_2}, $pbest_p$ of classical PSO. In the same way, the restrictors parameters $\mathbf{v}_{p_{max}}$, $\mathbf{b}_p$ are respectively replaced for $\mathbf{v}_{q_{max}}$ and $\mathbf{b}_q$ in the swarm of robots. Worth mentioning that these parameters are intrinsically related to the characteristics of the robots and to problem specificity.

The parameters w_q, $\mathbf{v}_{q_{max}}$ and $\mathbf{b}_q$ depends on the kind of robots and the physical parameters related to environment. According to kind of the robot, that parameter can vary significantly.

As we are using different sets of particles acting in distinct environments, we use different fitness functions for each set. For virtual particles and robots we use f and h as fitness functions, respectively.

Such as classical PSO, the proposed technique returns a candidate solution associated with a member of the swarm. In this case, this member can be a virtual particle or a robot depending on value given by respectively fitness function. For minimization problem, the returned value is the lower met by a candidate, regardless of it be a robot or a virtual particle.

In real-life application, if the solution is given by a virtual particle, it is necessary to send a robot to that position to verify the solution found.

In order to verify a possible modification occurred in the original map, the fitness function of particles q are evaluated with older estimator for environment (f) and compared to fitness obtained by the robots using h function. If there is a difference among the results evaluated, we make a mark on map and position x_q and the value of fitness function evaluated with h are stored in the object map respectively in $map.position$ and $map.value$ attributes.

With this mark, we can, for example, make a update in the original map and, in the next time, the updated map can be used instead the old one. The object map is implemented to provide a indication of the changed regions in the environment. In real-life application, for example in search and rescue missions, this indication can be used by the manager of the mission or the stakeholders to take decisions and coordinate the teams involved in the mission.

The steps to implement the PSO Algorithm with virtual particles and robots are shown in Algorithm 3.

Algorithm 3: Hybrid PSO Pseudo-code

```
1  Initialize parameter of swarm of robots;
2  Initialize virtual swarm/particles parameters;
3  Initialize robots in real environment;
4  $BindProcess(Q)$;
5  Initialize virtual particles in the virtual search space;
6  gbest $\leftarrow f(\mathbf{x}_0)$;
7  $i \leftarrow 0$;
8  while stop condition not satisfied do
9      foreach virtual particle $p$ do
10         Generate the random numbers $\mathbf{r}_1$ and $\mathbf{r}_2$;
11         Update velocity $\mathbf{v}_p$ according to Equation 2;
12         Update position $\mathbf{x}_p$ according to Equation 1;
13         Evaluate fitness $f(\mathbf{x}_p)$;
14         Update pbest$_p$ and gbest ;
15         if robot paired then
16             Generate the random numbers $\mathbf{r}_a$ and $\mathbf{r}_b$;
17             Update velocity $\mathbf{v}_q$ according to Equation 5;
18             Update position $\mathbf{x}_q$ according to Equation 4;
19             Evaluate fitness $h(\mathbf{x}_p)$;
20             Update pbest$_q$ and gbest ;
21             if $f(\boldsymbol{x}_p)! = h(\boldsymbol{x}_q)$ then
22                 Mark on/Update original map;
23                 $map.position[i] \leftarrow \mathbf{x}_q$;
24                 $map.value[i] \leftarrow h(\mathbf{x}_q)$;
25                 $i \leftarrow i + 1$;
26                 $\mathbf{x}_p \leftarrow \mathbf{x}_q$;
27                 $\mathbf{v}_p \leftarrow \mathbf{v}_q$;
28             end
29         end
30     end
31 end
32 if gbest is obtained from $p$ group then
33     return $f(\boldsymbol{gbest})$
34 end
35 else
36     return $h(\boldsymbol{gbest})$
37 end
```

IV. Computational Experiments and Results

In order to measure the performance of the proposed method some computational experiments are executed. We set a swarm with 55 virtual particles (p=55) and 5 robots (q=5), therefore, with a total of 50 agents because the binding process. As parameters for hybrid PSO are used c_{p_1}=2.05, c_{p_2}=2.05, w_p=0.5, c_{q_1}=2.05, c_{q_2}=2.05, w_q=0.5. The restrictor parameters for each dimension are $\mathbf{v}_{p_{max}}$=[-1,1], $\mathbf{b}_p$=[-5.12, 5.12], $\mathbf{v}_{q_{max}}$=[-1,1], $\mathbf{b}_q$=[-5.12, 5.12]. We compare the proposed method with classical PSO executed in two ways: with the same number of virtual particles that our proposal (55 particles) and with the same number of real particles that our proposal (5 particles). Along this paper, the first way will be referenced as Large Swarm, second one as Reduced Swarm and our proposal as Hybrid Swarm. For each execution, the particles are initialized in random positions. These initialization positions are same for all three algorithms' executions.

After tests, we observe that with 1000 iteration the method converged. Even we are dealing with different population sizes, we observed that all methods stabilized after about the same iteration interval and don't improve from then onward. We use the number of iterations as stop criterion instead of number of function evaluations because of the stabilization observed. Thus, as stop condition it was used the number of iteration, $iter_{max}$=1000. We named as simulation each set of execution of the algorithms.

As explained in previous session, the proposed technique need a representation of the environment in original state (f) and a representation for the degraded (altered) environment state (h). To test the proposed technique, we create six different scenarios through the change of functions f and h.

For each scenario, the function f that represent the original environment is altered according to Equation 3 to generate a modified environment represented by h. Table I shows the tests scenarios with the respective original environment representation (f) and the environment modifier applied (g). The scenarios are separated in two kinds for better inspection: Sphere scenarios and Rastrigin scenarios. The Sphere scenarios are SC-1, SC-2, SC-3 and SC-4. The Rastrigin scenarios are SC-4 and SC-5.

TABLE I
Test Scenarios

Scenario	Original Environment f	Environment Modifier g
SC-1	Sphere	Rastrigin
SC-2	Sphere	Ellipse
SC-3	Sphere	Rosenbrock
SC-4	Sphere	Ackley
SC-5	Rastrigin	Ellipse
SC-6	Rastrigin	Ackley

The benchmark functions used in the tests scenarios are based on IEEE Congress on Evolutionary Computation (CEC) functions [12]. These functions are employ to compose the objective function in each scenario. For all scenarios we have a minimization problem, thus the algorithms looking for the minimum value of the objective functions.

All environment modifiers are apply centered in origin and with action radius $\mathbf{R}$. This radius is used to control the level of changes in environment and is defined for each D dimension, that is, $\mathbf{R} = \{R_1, R_2, R_3, ..., R_D\}$. Radius $\mathbf{R}$ can be interpreted as a limiter of the g function domain.

For all scenarios we used ten dimension (D=10) and actuation radius of degradation $\mathbf{R}$ is set to 1.0 ($\mathbf{R}$=1.0).

To compare the three algorithms, for each scenario, we executed 50 simulations and the average convergences curves were plotted. Figures 2, 3, 4 and 5 show the convergences for scenarios SC-1, SC-2, SC-3 and SC-4 (Sphere scenarios). Figures 6 and 7 show the convergences for scenarios SC-5 and SC-6 (Rastrigin scenarios). Our proposal and the Large Swarm exhibited the same behavior and both converge to **gbest** with similar values. Both presented better results than Reduced Swarm for all test scenarios.

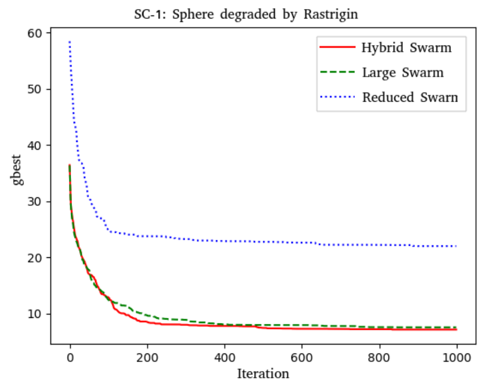

Fig. 2. Average convergence for 50 simulations in scenario SC-1

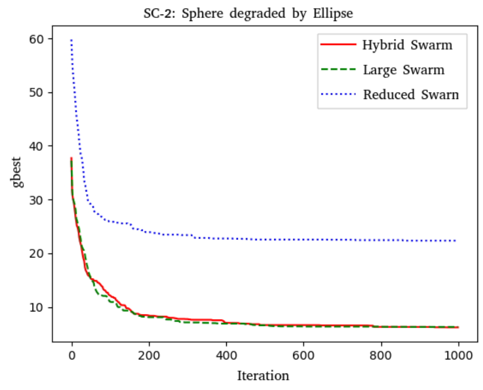

Fig. 3. Average convergence for 50 simulations in scenario SC-2

For all Scenarios our proposal (Hybrid Swarm) achieves equivalent results than Large Swarm and superior results than Reduced Swarm. The Figure 8 shows the boxplots for Sphere scenarios (SC-1, SC-2, SC-3, SC-4) and Figure 9 for Rastritin scenarios (SC-5, SC-6). We observe that the behavior of the proposed technique is kept in all tests. These results show that the proposed technique can improve search capabilities of a small swarm of robots.

We executed tests with the scenarios shown in Table I varying the number of dimensions D to verify the behavior of the proposed technique. We use 20, 30, 50 and 100 dimensions in the tests. In all tests the general behavior was kept, that is, the swarm assisted by virtual particles (Hybrid Swarm) achieves results close to Large Swarm and the Reduced Swarm achieves the worst results.

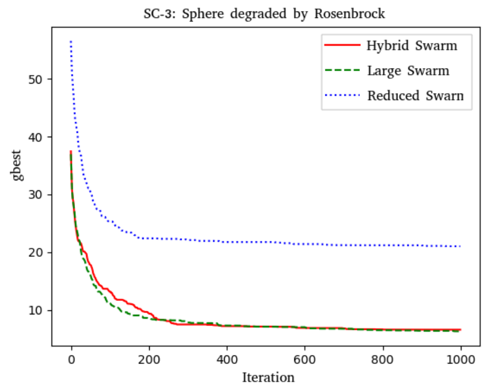

Fig. 4. Average convergence for 50 simulations in scenario SC-3

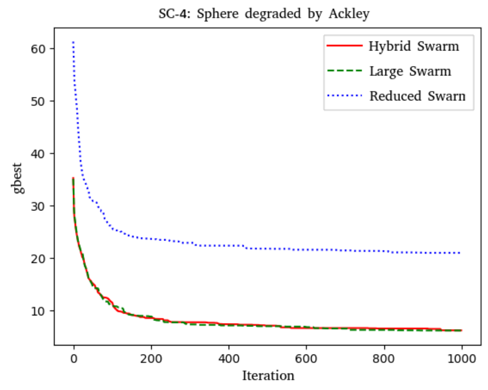

Fig. 5. Average convergence for 50 simulations in scenario SC-4

This kind of test is important to assess the robustness of the proposal. For robots application, the dimensions of the problem can represent, beyond the environment itself, the state of the robot. Each dimension can map a variable of the robot such as a sensor, an actuator or internal informations. Some examples of that variables are battery level, position in a coordinate system, orientation, communication signal level, position of the actuators, internal memory usage, internal diagnostic indication.

V. Conclusions

The computational results show that is possible improve the search capabilities of a small swarm of robots through the usage of virtual entities. This proposal achieves similar performance of a large swarm of robot, however with less costs by using less robots. Furthermore, the technique demonstrates be robust for application in large dimension problems.

As future works, we intent to assess the performance of the proposed method in different environments to investigate its robustness; investigate the influence of parameters of virtual swarm in the convergence of the method. Implementing in

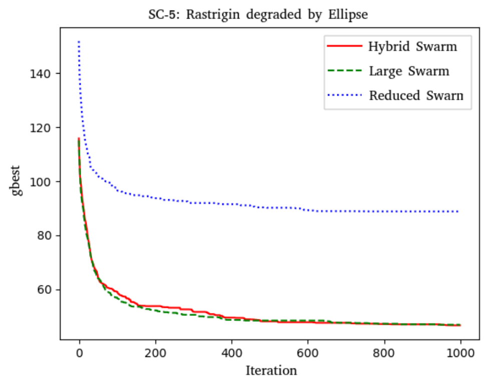

Fig. 6. Average convergence for 50 simulations in scenario SC-5

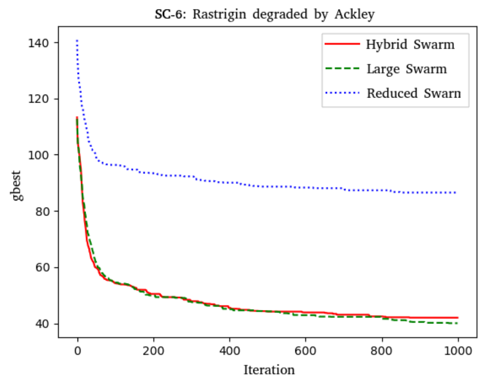

Fig. 7. Average convergence for 50 simulations in scenario SC-6

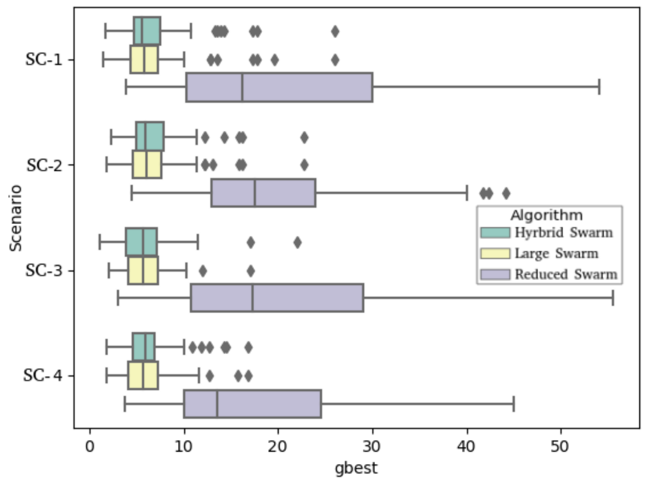

Fig. 8. Boxplots comparing the algorithms in Sphere scenarios SC-1, SC-2, SC-3 and SC-4.

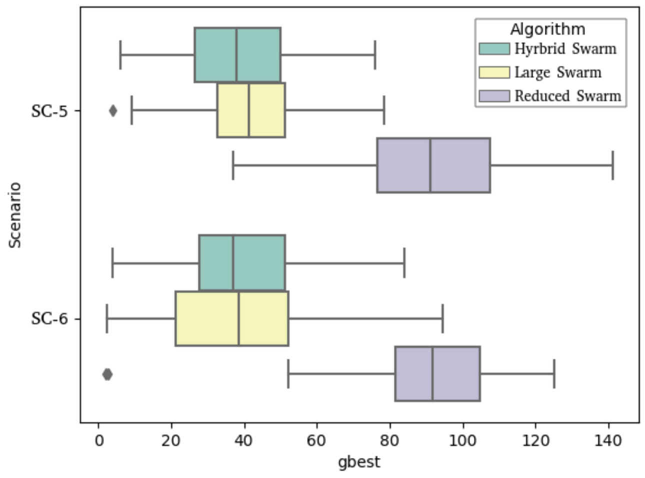

Fig. 9. Boxplots comparing the algorithms in Sphere scenarios SC-5 and SC-6.

virtual simulators for robotics are interesting to validate the technique with robots in a real environment.

Acknowledgment

The authors would like to thank the support from FACEPE (IBPG-0123-3.04/18) and PPGES/UPE.

References

[1] K. Nagatani, H. Ishida, S. Yamanaka, and Y. Tanaka, "Three-dimensional localization and mapping for mobile robot in disaster environments," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 4. IEEE, 2003, pp. 3112–3117.

[2] S. C. Nair, E. M. Coronado, M. T. Frye, and Y. Qin, "Swarm intelligence for the control of a group of robots," in *2015 10th System of Systems Engineering Conference (SoSE)*. IEEE, 2015, pp. 205–207.

[3] U. Goel, S. Varshney, A. Jain, S. Maheshwari, and A. Shukla, "Three dimensional path planning for uavs in dynamic environment using glow-worm swarm optimization," *Procedia computer science*, vol. 133, pp. 230–239, 2018.

[4] F. Yang, P. Wang, Y. Zhang, L. Zheng, and J. Lu, "Survey of swarm intelligence optimization algorithms," in *2017 IEEE International Conference on Unmanned Systems (ICUS)*. IEEE, 2017, pp. 544–549.

[5] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[6] M. Senanayake, I. Senthooran, J. C. Barca, H. Chung, J. Kamruzzaman, and M. Murshed, "Search and tracking algorithms for swarms of robots: A survey," *Robotics and Autonomous Systems*, vol. 75, pp. 422–434, 2016.

[7] C. Greenhagen, T. Krentz, J. Wigal, and S. Khorbotly, "A real-life robotic application of the particle swarm optimization algorithm," in *2016 Swarm/Human Blended Intelligence Workshop (SHBI)*. IEEE, 2016, pp. 1–5.

[8] J. Kennedy and R. Eberhart, "Particle swarm optimization (pso)," in *Proc. IEEE International Conference on Neural Networks, Perth, Australia*, 1995, pp. 1942–1948.

[9] J. Kennedy, "The particle swarm: social adaptation of knowledge," in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*. IEEE, 1997, pp. 303–308.

[10] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*. IEEE, 1998, pp. 69–73.

[11] J. C. Bansal, P. Singh, M. Saraswat, A. Verma, S. S. Jadon, and A. Abraham, "Inertia weight strategies in particle swarm optimization," in *2011 Third world congress on nature and biologically inspired computing*. IEEE, 2011, pp. 633–640.

[12] X. Li, K. Tang, M. N. Omidvar, Z. Yang, K. Qin, and H. China, "Benchmark functions for the cec 2013 special session and competition on large-scale global optimization," *gene*, vol. 7, no. 33, p. 8, 2013.