Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

A COMPARISON ANALYSIS OF SWARM INTELLIGENCE ALGORITHMS FOR ROBOT SWARM LEARNING

Jiaqi Fan

Department of Information Management
Donghua University
1882 W. Yanan Rd.
Shanghai, 200051, P.R. CHINA

Mengqi Hu

Department of Mechanical and Industrial Engineering
University of Illinois at Chicago
842 W. Taylor St.
Chicago, IL 60607, USA

Xianghua Chu

Department of Management Science
Shenzhen University
3688 Nanhai Ave.
Shenzhen 518060, P.R. CHINA

Dong Yang

Department of Information Management
Donghua University
1882 W. Yanan Rd.
Shanghai, 200051, P.R. CHINA

ABSTRACT

In the robot swarm, each robot can freely form swarm with others to share information. Although particle swarm optimization (PSO) has been demonstrated to outperform Q-learning and evolutionary algorithms, less study is conducted to characterize various swarm intelligence (SI) algorithms and evaluate their performances for robot swarm learning. In this research, we select three representative SI algorithms include bat algorithm (BA), PSO, grey wolf optimizer (GWO) according to their learning strategies. These three algorithms are implemented in a distributed manner and compared under various number of robots (NR) and communication ranges (CR). The simulation results demonstrate that: 1) PSO outperforms BA and BA outperforms GWO in general, 2) GWO performs better than PSO and BA under large NR and long CR, 3) increasing NR and CR can significantly improve the performance of GWO. These results can shed lights on selection of optimal SI algorithms for robot swarm under various scenarios.

1 INTRODUCTION

A group of intelligent robots (a.k.a. robot swarm) which can be self-organized to form swarms to execute complex tasks have attracted great attention recently (Tan and Zheng 2013). Increasingly sophisticated intelligent algorithms are needed to optimally control the robot swarm (Pugh and Martinoli 2006). In the past few years, several exciting approaches are being explored including supervised and unsupervised learning algorithms (Pugh and Martinoli 2009a). Comparing with supervised learning approach, the unsupervised learning algorithm is more noise-tolerant and fast-response (Pugh and Martinoli 2009a).

In this research, we mainly focus on unsupervised learning based controller design for robot swarm. To develop efficient unsupervised learning algorithms, the machining learning, evolutionary computation, and swarm intelligent algorithms were explored. Two variants of Q-learning algorithms were developed including Q-learning with discrete and continuous state space (Di Mario et al. 2013). Q-learning is a model-free reinforcement learning approach. It was demonstrated that Q-learning with continuous state space outperforms the Q-learning with discrete state space for robot swarm control. Genetic algorithm (GA) which is an evolutionary algorithm was studied to optimize an artificial neural network (ANN) controller (Pugh and Martinoli 2007, Pugh and Martinoli 2009b) for robot swarm. Particle swarm optimization (PSO), a powerful swarm intelligence algorithm, was employed to study the robot swarm learning recently (Pugh and Martinoli 2006, Pugh and Martinoli 2007, Pugh and Martinoli 2009a). For example, a distributed version of PSO algorithm was proposed to improve the ANN controller performance in terms of computational cost and robustness (Pugh and Martinoli 2009b, Di Mario and Martinoli 2014). To enable robots fast-response capabilities to noises, several noise-tolerant PSO algorithms were developed (Di Mario et al. 2015, Navarro et al. 2016). It was demonstrated that the PSO based learning algorithm outperforms GA (Pugh and Martinoli 2006, Di Mario and Martinoli 2014) and Q-learning (Di Mario et al. 2013).

The term swarm intelligence was first introduced in the early 1990's from research on a cellular robotic system (Beni and Wang 1993). Ever since, a wide variety of swarm intelligence algorithms have been proposed (Parpinelli and Lopes 2011). For example, ant colony optimization algorithm is inspired from the ants behavior in finding the shortest path between food and their nest (Colorni et al. 1991). Artificial bee colony (Karaboga and Akay 2009) mimics the food searching process of the honey bee swarm. The bat algorithm was inspired by the echolocation behavior of bats (Yang 2010).

To the best of our knowledge, less study was conducted to compare the performance of various swarm intelligence algorithms for robot swarm learning which can shed lights on the selection of optimal swarm intelligence algorithms for robot swarm under various scenarios. To this end, we will characterize the swarm intelligence algorithms based on their learning strategies (e.g., social learning, cognitive learning, naive learning) and select three representative algorithms for comparison analysis. A distributed version of each algorithm will be implemented where each robot can evolve one or multiple solutions simultaneously. Through mimicking the individual and swarm behaviors in the swarm intelligence algorithm, we will compare the performance of various learning strategies for robot swarm learning. The obstacle avoidance benchmark task (Pugh and Martinoli 2006, Pugh and Martinoli 2007) is employed to evaluate the algorithm performance. To further analyze the sensitivity of each algorithm, we will study various number of robots and communication ranges. The performances of each algorithm will be statistically compared. The research findings can provide guidelines to select appropriate learning approach for the robot swarm. For example, what are the scenarios to select social learning? When to select cognitive learning?

This paper is organized as follows: background knowledge on swarm intelligence algorithms is introduced in Section 2, Section 3 presents the proposed simulation model; followed by the comparison experiments in Section 4, and conclusions and future research are drawn in Section 5.

2 SWARM INTELLIGENCE ALGORITHMS REVIEW

Extensive research efforts have been devoted to developing swarm intelligence algorithms since 1990. In this research, the swarm intelligence algorithms are characterized based on their learning behaviors which include social learning, cognitive learning, and naive learning. The social learning is defined as the individual's capability to learn from others in the swarm. The social learning requires good communication among robots in the swarm. The cognitive learning is the capability for each individual to learn from its previous experience. The individuals employ a random wandering/flying without specific guidance in the naive learning. In this exploring study, we will compare the performances of these three learning approaches using robot swarm under various scenarios (e.g., different communication range, various numbers of robots). We study three representative swarm intelligence algorithms which are: 1) PSO (Shi and Eberhart 1998) combing both social and cognitive learning, 2) bat algorithm (BA) (Yang 2010) employing both social and naive learning, and 3) grey wolf optimizer (GWO) (Mirjalili et al. 2014) only adopting social learning.

2.1 Particle Swarm Optimization

PSO is a swarm intelligence developed to mimic a flock of birds that communicate together as they fly (Eberhart and Kennedy 1995, Kennedy and Eberhart 1995). Similar to most population-based algorithms, a group of individuals termed as a swarm is created to search on the solution space. Each individual is defined as a particle whose motion is directed by three components: 1) its current velocity; 2) its best position (pBest) found so far; 3) the best position (gBest) found so far from its neighbors where the neighborhood is defined by the topology.

In the PSO with inertia weight, the velocity ($\mathbf{v}_p^i$) and position ($\mathbf{x}_p^i$) for particle p at iteration i are updated as (Shi and Eberhart 1998),

$$\mathbf{x}_p^{i+1} = \mathbf{x}_p^i + \mathbf{v}_p^{i+1} = \mathbf{x}_p^i + w \times \mathbf{v}_p^i + c_1 \times r_{1,p}^i \times (\mathbf{p}_p^i - \mathbf{x}_p^i) + c_2 \times r_{2,p}^i \times (\mathbf{p}_g^i - \mathbf{x}_p^i), \tag{1}$$

where D-dimensional vectors $\mathbf{v}_p^i$ and $\mathbf{x}_p^i$ are the velocity and position of the p^{th} particle ($\mathbf{v}_p^i \in [-\mathbf{V}_{max}, +\mathbf{V}_{max}]$) respectively, $\mathbf{V}_{max}$ is used to constrain the velocity for each particle and is usually set between 0.1 and 1.0 times the search range of the solution space (Banks et al. 2007); $\mathbf{p}_p^i$ is the best position found so far by the p^{th} particle; $\mathbf{p}_g^i$ is the best position found so far by the swarm; $r_{1,p}^i$ and $r_{2,p}^i$ represent two independent random numbers uniformly distributed on [0, 1]; c_1 is the cognitive learning factor which represents the attraction that a particle has toward its own success $\mathbf{p}_p^i$; c_2 is the social learning factor which represents the attraction that a particle has toward the swarm's best position $\mathbf{p}_g^i$; w is the inertia weight.

2.2 Bat Algorithm

Inspired by the echolocation behavior of bats, BA was presented in (Yang 2010). In BA, there are three basic rules: (1) all bats employ echolocation to measure distance, recognize prey/food and avoid obstacles; (2) each bat flies with a velocity using a fixed frequency f_{min}, varying wavelength λ and loudness A_0 to adjust the flying for exploration and exploitation; (3) the loudness decreases from A_0 to A_{min}. The wavelength can be fixed in certain implementations (Yang 2010); the rate of pulse emission r_i works as a threshold between [0, 1], where 0 refers to no pulse emission, and 1 means the largest rate of pulse emission.

In BA, bats move to a new position by adjusting their velocities through acquiring knowledge from the global best information. The velocity ($\mathbf{v}_p^i$) and position ($\mathbf{x}_p^i$) for bat p at iteration i are updated as

$$\mathbf{x}_p^{i+1} = \mathbf{x}_p^i + \mathbf{v}_p^{i+1} = \mathbf{x}_p^i + \mathbf{v}_p^i + (\mathbf{x}_p^i - \mathbf{x}_*)\left[f_{min} + (f_{max} - f_{min})\beta\right], \tag{2}$$

where $\mathbf{x}_*$ is the global best information obtained so far, β is a random number uniformly distributed on [0, 1], f_{max} and f_{min} are two parameters depending on the studied problem. The learning strategy of BA is similar to PSO but without cognitive learning. Once a promising area is found, the information is shared in a timely manner for guidance to the other bats in the population. In principal, the pulse emission rate (r_i) and the loudness (A_i) are used to adjust the probability of the converging operation. The pulse emission rate determines the probability to produce new solutions near the best positions while the loudness decides the possibility to accept the new solutions. Both of them are updated only if the solutions are improved. When the condition $rand > r_i$ satisfied where $rand$ is a random number uniformly distributed on [0, 1], a naive learning strategy is implemented where a new solution is generated using random walk

$$\mathbf{x}_p^{i+1} = \mathbf{x}_* + \Delta, \tag{3}$$

where Δ is a random step size.

2.3 Grey Wolf Optimizer

GWO imitates the leadership hierarchy of grey wolves, and models four different roles comprising of an alpha, beta, delta and omega ranking from the highest level to the lowest level (Mirjalili et al. 2014). These four types of roles are characterized by a strict social dominant hierarchy in grey wolves. Moreover, group hunting, an interesting social activity of grey wolves, including encircling prey, hunting, attacking and searching is considered in the GWO. Since there is a strict social dominant hierarchy imposed, most of the members act as followers (omega) of the dominators (alpha, beta, and delta) who rank in the first three places according to performance. In GWO, the roles of alpha, beta and delta are dynamically changed based on their current position in each iteration. Social learning is adopted in GWO, and the position ($\mathbf{x}_p^i$) for wolve p at iteration i are updated as

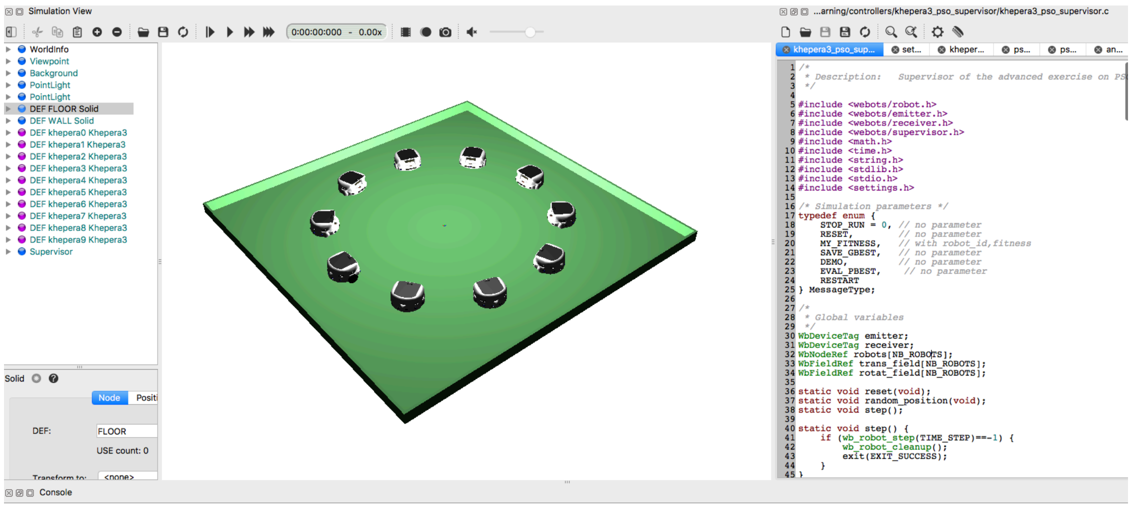
$$\mathbf{x}_p^{i+1} = \sum\nolimits_{L \in \alpha,\beta,\delta} \mathbf{x}_L / 3, \tag{4}$$

$$\mathbf{x}_L = \mathbf{x}_L^i - (2 \times a \times \mathbf{r}_1 - a) \times |2 \times \mathbf{r}_2 \times \mathbf{x}_L^i - \mathbf{x}_p^i|, L \in \alpha,\beta,\delta, \tag{5}$$

where $\mathbf{x}_\alpha^i$, $\mathbf{x}_\beta^i$, and $\mathbf{x}_\delta^i$ refer to the leaders' positions (alpha, beta, and delta correspond to the first, the second and the third best individual separately), a is a weight linearly decreasing from 2 to 0 over the generations, $\mathbf{r}_1$ and $\mathbf{r}_2$ are random vectors uniformly distributed on [0, 1]. It can be observed from that each individual simultaneously and evenly learns from alpha, beta and delta. Learning from the leaders is not biased toward any of the individuals. It also reveals that grey wolves as a group animal always move together towards their prey. All individuals keep living in a pack and acting simultaneously when exploiting and exploring. Although there is a strict social hierarchy in population, the members collaborate closely to search, encircle, and hunt for food.

3 ROBOT SWARM SIMULATION MODEL

In this research, a swarm of Khepera III mobile robots (Di Mario and Martinoli 2014) is studied using Webots (Michel 2004) which is a mobile robotics simulation software to model, program and simulate mobile robots. Khepera III mobile robot is a differential wheeled vehicle which can detect short and medium range obstacles using nine infra-red sensors and five ultrasound sensors (Di Mario and Martinoli 2014). In the simulation model, a swarm of robots is distributed and navigated in a square arena (see Figure 1). For each robot, the walls and other robots are considered as obstacles.



```c
/*
 * Description:   Supervisor of the advanced exercise on PS
 */

#include <webots/robot.h>
#include <webots/emitter.h>
#include <webots/receiver.h>
#include <webots/supervisor.h>
#include <math.h>
#include <time.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <settings.h>

/* Simulation parameters */
typedef enum {
    STOP_RUN = 0, // no parameter
    RESET,        // no parameter
    MY_FITNESS,   // with robot_id,fitness
    SAVE_GBEST,   // no parameter
    DEMO,         // no parameter
    EVAL_PBEST,    // no parameter
    RESTART
} MessageType;

/*
 * Global variables
 */
WbDeviceTag emitter;
WbDeviceTag receiver;
WbNodeRef robots[NB_ROBOTS];
WbFieldRef trans_field[NB_ROBOTS];
WbFieldRef rotat_field[NB_ROBOTS];

static void reset(void);
static void random_position(void);
static void step();

static void step() {
    if (wb_robot_step(TIME_STEP)==-1) {
        wb_robot_cleanup();
        exit(EXIT_SUCCESS);
    }
}
```

Figure 1: Robot swarm simulation model in Webots.

The obstacle avoidance benchmark task studied in (Pugh and Martinoli 2006, Pugh and Martinoli 2007) is adopted to evaluate the algorithm performance. The fitness function used in the swarm intelligent algorithm is defined as

$$F = V \times (1 - \sqrt{\Delta v}) \times (1 - i), \tag{6}$$

where V is the average left and right wheel speed, Δv is the speed difference for left and right wheels, i is the proximity sensor activation value of the most active sensor. All the three quantities, V, Δv, and i are normalized to the interval [0, 1]. The robot which moves fast, turns as little as possible, and spends as little time as possible close to obstacles will have large value of F (Di Mario and Martinoli 2014).

Each robot will run a same ANN based controller with different weights for the ANN. In the ANN based controller, the input layer has 13 neurons to accept inputs for the nine infrared sensors, two constant biases, the previous speed of left and right wheels. The hidden layer is designed to include two neurons and the output layer has two neurons to control the left and right wheels of the robot. The readers can refer to (Pugh and Martinoli 2006, Pugh and Martinoli 2007) for details about the ANN based controller and multi-robot simulation model. The weights will be evolved using the swarm intelligent algorithms which means an optimization problem with 30-dimensional search space is solved to maximize the fitness function defined in Eq. (6).

4 SIMULATION RESULT ANALYSIS

Let *NR* and *CR* represent the number of robots and communication range, respectively. We will compare the performance of these three algorithms (BA, GWO, and PSO) by changing *NR* from 4, 10 to 20, and *CR* from 0.3, 0.8 to 1.5 meters. The arena size is defined as 1.5 meters. Each robot is assumed to only communicate with other robots in its neighborhood with distance less than *CR*. This indicates that the robot can communicate with more robots when the *CR* is large. To guarantee the fair comparison, the population size for each algorithm is set as 20 which means each robot will maintain 5 solutions when *NR* is 4, maintain 2 solutions when *NR* is 10, and maintain 1 solution when *NR* is 20. Large value of *NR* will make the arena crowded. The maximum number of iterations for each algorithm is set as 50, and each algorithm is independently run 30 times. The fitness values for each run is recorded.

4.1 Comparison Analysis for the Three Algorithms under Settings of *NR*=10 and *CR*=0.8

In this set of experiments, we set *NR* as 10 and *CR* as 0.8 meter. The average, standard deviation, maximum, and minimum values for the best fitness obtained at each run are presented in Table 1. It is observed that PSO performs the best among these three algorithms, and BA performs better than GWO. The best fitness value is obtained by the BA, and the performance of GWO is more stable comparing to BA and PSO.

Table 1: Experimental results for the three algorithms (*NR*=10, *CR*=0.8).

	Average	**Standard Deviation**	**Maximum**	**Minimum**
BA	0.7579	0.1170	0.9604	0.5066
GWO	0.6212	0.0893	0.8363	0.4974
PSO	0.8037	0.1173	0.9407	0.5536

To further demonstrate the differences among these three algorithms, we rank the 30 runs for each algorithm and present the convergence curve for the top 15^{th} run in Figure 2. As we observe, the GWO has premature convergence issues. Although the GWO has very good initial solution, it is stuck in the initial solution for more than 20 iterations. BA does not have improvement after 30 iterations. PSO can maintain a good population diversity to improve solution quality. According to the statistical t-test, we conclude that PSO outperforms BA, and both PSO and BA significantly outperform GWO.

4.2 Statistical Comparison of the Three Algorithms under Various Settings of *NR* and *CR*

In this set of experiments, the performances of these three algorithms are compared under different values of *NR* and *CR*. The box plots for these three algorithms using 4, 10, and 20 robots are shown in Figures 3-5 where the mean value over the 30 runs is represented using triangle and the outlier is represented using circle. Please note “CR0.3” represents that *CR*=0.3. In general, we conclude that PSO is better than BA, and BA is better than GWO when *NR* is 4 and 10. However, GWO outperforms BA when *NR* is 20 and *CR* is 0.8 and 1.5, and outperforms PSO when *NR* is 20 and *CR* is 1.5.

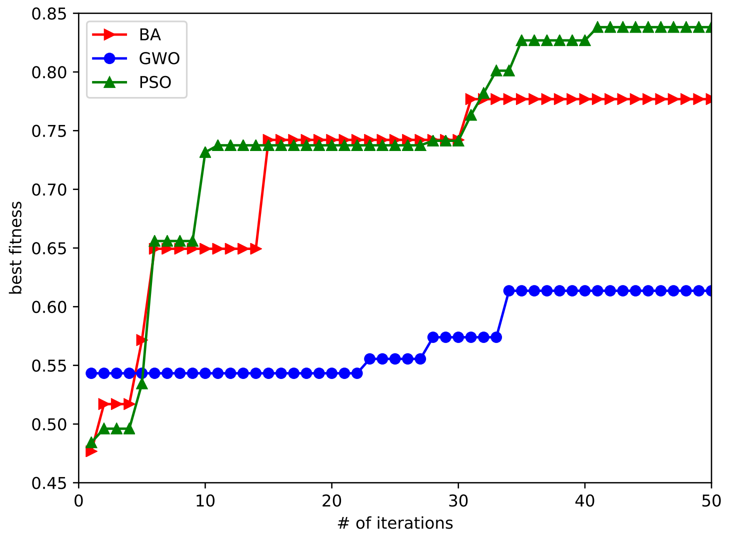


Figure 2: Convergence curve for BA, GWO and PSO (*NR*=10, *CR*=0.8).

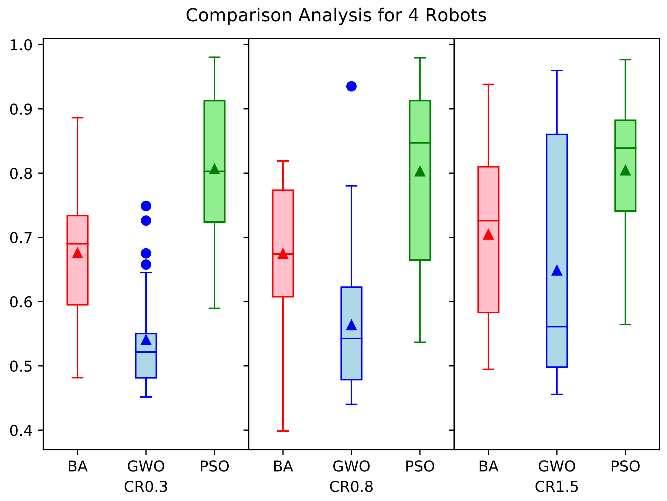


Figure 3: Box plots for BA, GWO and PSO (*NR*=4).

The statistical t-test results for these three algorithms are presented in Table 2 where the algorithms are ranked based on the t-test results. The algorithm A will be assigned symbols (“$>>$”, “$>$”, or “=”) based on the following rules: 1) symbol “$>>$” if it is significantly better than algorithm B (e.g., p-value < 0.05), 2) symbol “$>$” if it is better than algorithm B (e.g., p-value is in the range [0.05, 0.90]), 3) symbol “=” if it performs the same as algorithm B (e.g., p-value > 0.90). The two symbols of the rank #1 algorithm indicate its relationship between the rank #2 and rank #3 algorithms, and the symbol of the rank #2 algorithm indicates its relationship with the rank #3 algorithm. For example, GWO performs similar to PSO, and better than BA, and PSO performs better than BA when *NR*=20 and *CR*=1.5. We can conclude that: 1) the algorithm with both social and cognitive learning capabilities (e.g., PSO) can perform better than the algorithms without cognitive learning capability (e.g., BA and GWO), and 2) the algorithm only with social learning capability (e.g., GWO) may underperform the algorithms with hybrid learning strategies (e.g., BA and PSO) when *NR* is small and *CR* is short.

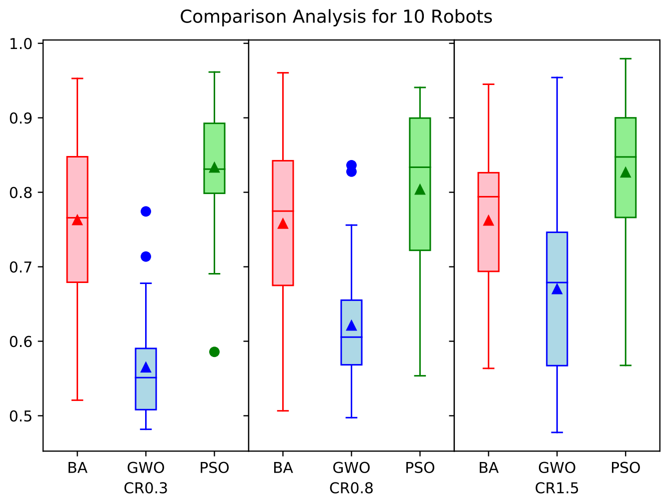


Figure 4: Box plots for BA, GWO and PSO (*NR*=10).

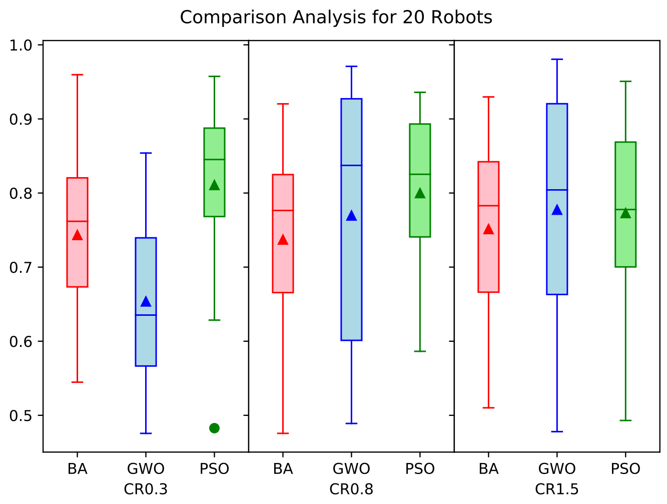


Figure 5: Box plots for BA, GWO and PSO (*NR*=20).

Table 2: Statistical t-test results for the three algorithms under various settings of *NR* and *CR*.

NR	***CR***	**#1**	**#2**	**#3**
4	0.3	PSO (>>, >>)	BA (>>)	GWO
4	0.8	PSO (>>, >>)	BA (>>)	GWO
4	1.5	PSO (>>, >>)	BA (>)	GWO
10	0.3	PSO (>>, >>)	BA (>>)	GWO
10	0.8	PSO (>, >>)	BA (>>)	GWO
10	1.5	PSO (>>, >>)	BA (>>)	GWO
20	0.3	PSO (>>, >>)	BA (>>)	GWO
20	0.8	PSO (>, >>)	GWO (>)	BA
20	1.5	GWO (=, >)	PSO (>)	BA

4.3 Statistical Comparison of Individual Algorithm under Various Settings of *NR*

In this set of experiments, the performances of each algorithm are compared under different values of *NR*. The box plots for these three algorithms are shown in Figures 6-8. Please note "Rob4" represents that *NR*=4. For BA, it performs the best when *NR*=10 and performs the worst when *NR*=4. However, BA under *NR*=10 performs slightly better than *NR*=20. The performances of GWO have significant improvement when *NR* is increasing. PSO performs very similar under various settings of *NR*.

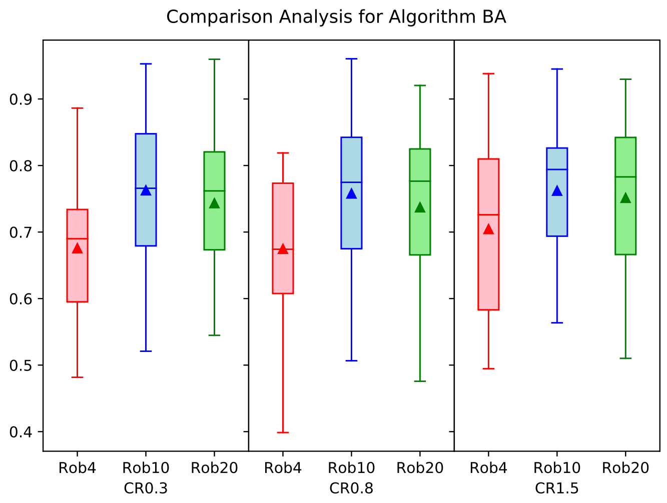


Figure 6: Box plots for BA under various settings of *NR*.

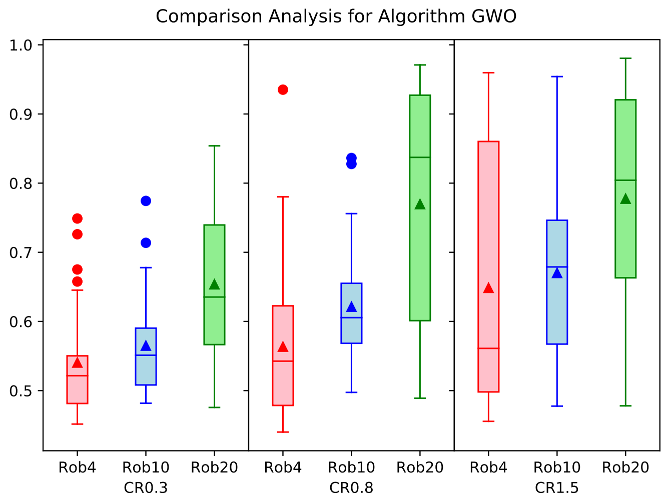


Figure 7: Box plots for GWO under various settings of *NR*.

The statistical *t*-test results for these three algorithms under various settings of *NR* are presented in Table 3. It is observed that GWO prefers larger number of robots. This is due to the fact that GWO just uses social learning to evolve the solutions, and increasing the number of robots can improve the social learning capability of each individual to better search the solution space. Both BA and PSO prefers a medium number of robots (e.g., 10) since smaller number of robots may weaken the social learning capability and larger number of robots may increase the density of the arena and thus to increase the collision possibility.

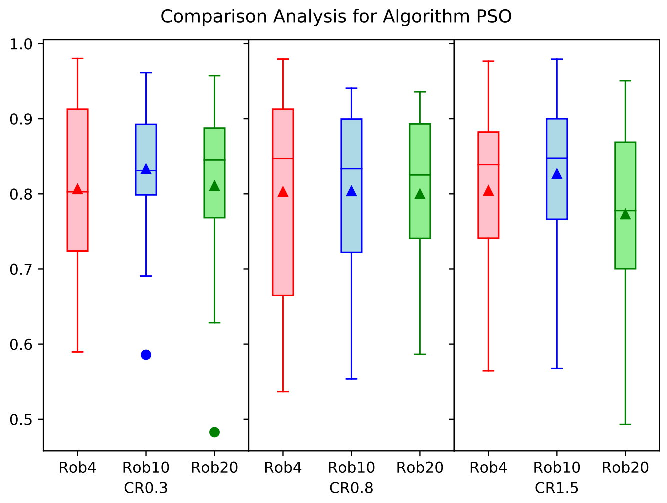


Figure 8: Box plots for PSO under various settings of *NR*.

Table 3: Statistical *t*-test results for individual algorithm under various settings of *NR*.

Algorithm	***CR***	**#1**	**#2**	**#3**
BA	0.3	10 (>, >>)	20 (>>)	4
BA	0.8	10 (>, >>)	20 (>>)	4
BA	1.5	10 (>, >)	20 (>)	4
GWO	0.3	20 (>>, >>)	10 (>)	4
GWO	0.8	20 (>>, >>)	10 (>>)	4
GWO	1.5	20 (>>, >>)	10 (>)	4
PSO	0.3	10 (>, >)	20 (>)	4
PSO	0.8	10 (=, >)	4 (=)	20
PSO	1.5	10 (>, >)	4 (>)	20

4.4 Statistical Comparison of Individual Algorithm under Various Settings of *CR*

In this set of experiments, the performances of each algorithm are compared under different values of *CR*. The box plots for these three algorithms are shown in Figures 9-11. The communication range does not have a significant impact to the performances of BA and PSO. The performances of GWO have significant improvement when *CR* is increasing.

Table 4: Statistical *t*-test results for individual algorithm under various settings of *CR*.

Algorithm	***NR***	**#1**	**#2**	**#3**
BA	4	1.5 (>, >)	0.3 (=)	0.8
BA	10	0.3 (=, >)	1.5 (>)	0.8
BA	20	1.5 (>, >)	0.3 (>)	0.8
GWO	4	1.5 (>>, >>)	0.8 (>)	0.3
GWO	10	1.5 (>, >>)	0.8 (>>)	0.3
GWO	20	1.5 (>, >>)	0.8 (>>)	0.3
PSO	4	0.3 (=, =)	1.5 (=)	0.8
PSO	10	0.3 (>, >)	1.5 (>)	0.8
PSO	20	0.3 (>, >)	0.8 (>)	1.5

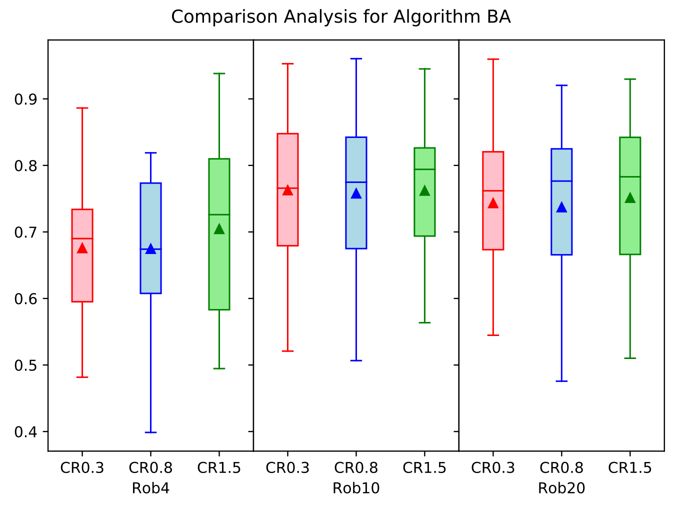


Figure 9: Box plots for BA under various settings of *CR*.

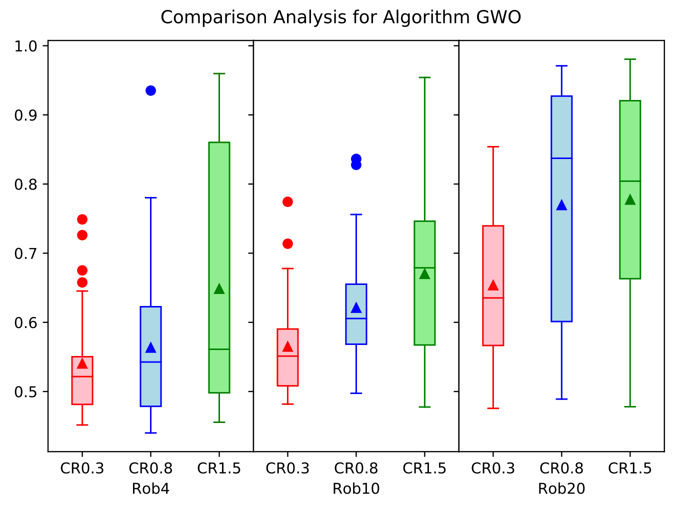


Figure 10: Box plots for GWO under various settings of *CR*.

The statistical *t*-test results for these three algorithms under various settings of *CR* are presented in Table 4. It is observed that BA slightly prefers large *CR* (e.g., 1.5 meters) and PSO slightly prefers small *CR* (e.g., 0.3 meter), but there does not have significant difference among various *CR* settings for BA and PSO. Large *CR* is preferred by GWO since it can increase the possibility of each robot to communicate with more robots and thus to improve the social learning capability.

5 CONCLUSION

In this research, we compare the performances of three representative swarm intelligence algorithms including BA, GWO and PSO for robot swarm learning. BA employs both social and naive learning strategies, GWO just implements social learning, and PSO adopts both social and cognitive learning strategies. The obstacle avoidance benchmark task is adopted to evaluate the algorithms' performance. A distributed version of each algorithm is implemented where each robot evolves one or more solutions simultaneously. These three algorithms are compared under various number of robots and communication ranges. It is observed from

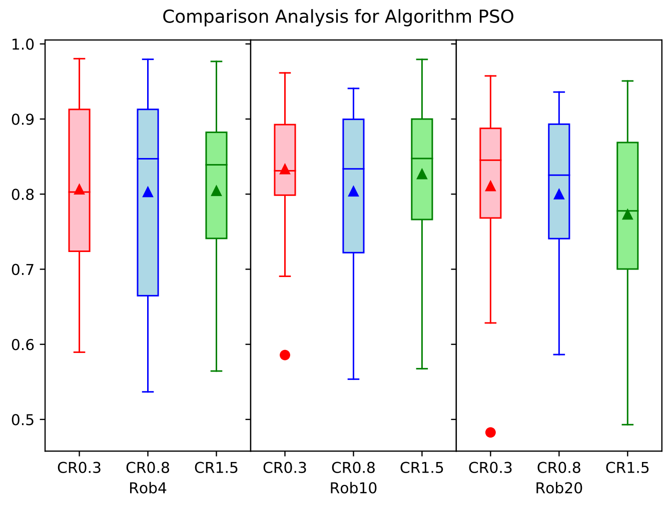


Figure 11: Box plots for PSO under various settings of *CR*.

the simulation results that: 1) the algorithm with hybrid learning strategies (e.g., PSO, BA) outperforms the algorithm with sole learning strategy (GWO), 2) increasing the number of robots and communication ranges can significantly improve the social learning capability of individuals in GWO and thus to improve the algorithm performance, 3) a medium number of robots are preferred by PSO and BA where small number of robots may decrease the social learning capability and large number of robots may increase the possibility for collision. These results can provide valuable insights to select appropriate swarm intelligence algorithm for various scenarios. For example, the PSO is preferred for robot swarm system with limited communication (e.g., underwater robot swarm) and small number of robots (e.g., expensive robotics). On the other side, the GWO is preferred when the communication range and number of robots are increased.

We will further evaluate the scalability and sensitivity of the swarm intelligence algorithms under various sizes of arena and maximum number of iterations. We will develop noise-tolerant swarm intelligence algorithms for robot swarm learning which consider various sources of uncertainties, such as sensor and actuator noise, and changes in the operation environment (Di Mario et al. 2015). A set of features will be identified to characterize the robot swarm (e.g., number of robots, communication range, etc.) and swarm intelligence algorithm (e.g., learning strategy, etc.), and a mapping model based on these features will be developed to optimally select swarm intelligence algorithms for robot swarm under various conditions.

REFERENCES

Banks, A., J. Vincent, and C. Anyakoha. 2007. “A Review of Particle Swarm Optimization. Part I: Background and Development”. *Natural Computing* 6 (4): 467–484.

Beni, G., and J. Wang. 1993. “Swarm Intelligence in Cellular Robotic Systems BT - Robots and Biological Systems: Towards a New Bionics?”. 703–712. Berlin: Springer Berlin Heidelberg.

Colorni, A., M. Dorigo, and V. Maniezzo. 1991. “Distributed Optimization by Ant Colonies”. In *1st Euro. Conf. Artif. Life*, 134–142.

Di Mario, E., and A. Martinoli. 2014. “Distributed Particle Swarm Optimization for Limited-Time Adaptation with Real Robots”. *Robotica* 32 (2): 193–208.

Di Mario, E., I. Navarro, and A. Martinoli. 2015. “A Distributed Noise-Resistant Particle Swarm Optimization Algorithm for High-Dimensional Multi-Robot Learning”. In *IEEE International Conference on Robotics and Automation*, 5970–5976.

Di Mario, E., Z. Talebpour, and A. Martinoli. 2013. “A Comparison of PSO and Reinforcement Learning for Multi-Robot Obstacle Avoidance”. In *IEEE Congress on Evolutionary Computation*, 149–156.

Eberhart, R., and J. Kennedy. 1995. “A New Optimizer using Particle Swarm Theory”. In *6th International Symposium on Micro Machine and Human Science*, 39–43.

Karaboga, D., and B. Akay. 2009. “A Survey: Algorithms Simulating Bee Swarm Intelligence”. *Artificial Intelligence Review* 31 (1-4): 61–85.

Kennedy, J., and R. Eberhart. 1995. “Particle Swarm Optimization”. In *IEEE International Conference on Neural Networks*, Volume 4, 1942–1948.

Michel, O. 2004. “Cyberbotics Ltd. Webots: Professional Mobile Robot Simulation”. *International Journal of Advanced Robotic Systems* 1 (1): 5.

Mirjalili, S., S. M. Mirjalili, and A. Lewis. 2014, mar. “Grey Wolf Optimizer”. *Advances in Engineering Software* 69:46–61.

Navarro, I., E. Di Mario, and A. Martinoli. 2016. “Noise-Resistant Particle Swarm Optimization for the Learning of Robust Obstacle Avoidance Controllers using a Depth Camera”. In *IEEE Congress on Evolutionary Computation*, 685–692.

Parpinelli, R. S., and H. S. Lopes. 2011, jan. “New Inspirations in Swarm Intelligence: a Survey”. *International Journal of Bio-Inspired Computation* 3 (1): 1–16.

Pugh, J., and A. Martinoli. 2006. “Multi-Robot Learning with Particle Swarm Optimization”. In *5th International Joint Conference on Autonomous Agents and Multiagent Systems*, 441–448. Hakodate, Japan.

Pugh, J., and A. Martinoli. 2007. “Inspiring and Modeling Multi-Robot Search with Particle Swarm Optimization”. In *IEEE Swarm Intelligence Symposium*, 332–339.

Pugh, J., and A. Martinoli. 2009a. “An Exploration of Online Parallel Learning in Heterogeneous Multi-Robot Swarms BT - Design and Control of Intelligent Robotic Systems”. 133–151. Berlin: Springer Berlin Heidelberg.

Pugh, J., and A. Martinoli. 2009b. “Distributed Scalable Multi-Robot Learning using Particle Swarm Optimization”. *Swarm Intelligence* 3 (3): 203–222.

Shi, Y., and R. Eberhart. 1998. “A Modified Particle Swarm Optimizer”. In *IEEE International Conference on Evolutionary Computation*, 69–73.

Tan, Y., and Z.-y. Zheng. 2013, mar. “Research Advance in Swarm Robotics”. *Defence Technology* 9 (1): 18–39.

Yang, X.-S. 2010. “A New Metaheuristic Bat-Inspired Algorithm BT - Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)”. 65–74. Berlin: Springer Berlin Heidelberg.

AUTHOR BIOGRAPHIES

JIAQI FAN is a master student in the Department of Information Management at Donghua University, China. Now, she is a visiting student in the Department of Mechanical and Industrial Engineering at the University of Illinois at Chicago. Her research interest is robot swarm learning. Her email address is jady_fan@163.com.

MENGQI HU is an Assistant Professor in the Department of Mechanical and Industrial Engineering at the University of Illinois at Chicago. His research interests include distributed decision making, swarm intelligence, and complex system engineering. His email address is mhu@uic.edu.

XIANGHUA CHU is an Assistant Professor with the Department of Management Science, College of Management, Shenzhen University, China. His current research interests include swarm intelligence, intelligent decision support, and optimization. His email address is x.chu@szu.edu.cn.

DONG YANG is a Professor in the Department of Information Management at Donghua University, China. His research interests include operational research, stochastic programming, and product configuration. His email address is yangdong@dhu.edu.cn.