Cooperative Area Extension of PSO
Transfer Learning vs. Uncertainty in a Simulated Swarm Robotics

Adham Atyabi[1] and David M. W. Powers[1,2]
[1]*CSEM Centre for Knowledge & Interaction Technology, Flinders University, Adelaide, South Australia*
[2]*Beijing Municipal Lab for Multimedia & Intelligent Software, Beijing University of Technology, Beijing, China*

Keywords: Swarm Robotics, Particle Swarm Optimization, Cooperative Learning, Transfer Learning, Knowledge Transfer.

Abstract: The study investigates the effectiveness of 2 variations of Particle Swarm Optimization (PSO) called Area Extended PSO (AEPSO) and Cooperative AEPSO (CAEPSO) in simulated robotic environments affected by a combinatorial noise. Knowledge Transfer, the use of the expertise and knowledge gained from previous experiments, can improve the robots decision making and reduce the number of wrong decisions in such uncertain environments. This study investigates the impact of transfer learning on robots' performance in such hostile environment. The results highlight the feasibility of CAEPSO to be used as the controller and decision maker of a swarm of robots in the simulated uncertain environment when gained expertise from past training is transferred to the robots in the testing phase.

1 INTRODUCTION

Navigation is the art of steering a course through a medium. Localization matches an actual position in the real world to a location inside a map. Planning is finding a short, collision-free path from the starting position towards the predefined ending location. As such, navigational techniques are useful and effective when the map on similar information is reliable. Nevertheless, in most real world application domains, the environment is dynamic, time-dependent, and uncertain. Such environmental conditions pose challenges to localization and map reliability. Under such circumstances, behavior-based approaches are suitable to address real world applications when engaged with navigation problems.

The Swarm Intelligence (SI) term, introduced by Beni, Hackwood, and Wang in 1989, used for systems in which, unsophisticated agents with collective behaviors have the capability of emerging to a global pattern by interacting locally with their environment. SI systems have the capability to solve the collective problems without centralized control. Particle Swarm Optimization (PSO), introduced by Kennedy and Eberhart in (Kennedy and Eberhart, 1995), has been inspired from animals' social behaviors which are illustrated by their social acts resulting in population survival. PSO is a self-adaptive population-based method in which, behaviors of the swarm are iteratively generated from the combination of social and cognitive behaviors of the swarm. A swarm can be imagined as consisting of members called particles. Particles cooperate with each other to achieve desired behaviors or goals. Particles' acts are governed based on simple local rules and interactions with the entire swarm. As an example, movement of a bird in a flock is based on adjusting movements with its flock mates (near by neighbors in the flock) (Sousa et al., 2004). Birds in a flock stay close to their neighbors and avoid collisions with each other. They do not take commands from any leader bird (there are no leader birds). This kind of social behavior (Swarm behavior) helps birds to achieve tasks such as protection from predators and searching for food (Grosan et al., 2006).

Although PSO has proved to be efficient in solving problems in various domains, it comes with considerable shortcomings. The shortcomings include premature convergence and difficulties with dynamic and real-world optimization. Enhanced versions of PSO called Area Extended PSO (AEPSO) and Cooperative AEPSO (CAEPSO) showed potential in dynamic and uncertain simulated environments (Atyabi et al., 2010). In the study, two variations of uncertainty (i.e. random noise and relational noise) are employed to assess the feasibility of the suggested approaches. The dynamically changing nature of the simulated en-

Atyabi A. and M. W. Powers D..
Cooperative Area Extension of PSO - Transfer Learning vs. Uncertainty in a Simulated Swarm Robotics.
DOI: 10.5220/0004456901770184
In *Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2013), pages 177-184
ISBN: 978-989-8565-70-9

vironment in (Atyabi et al., 2010) make it more difficult for the simulated swarm of simple robots to compensate the noise. CAEPSO tackled this problem through incorporating previously learned knowledge to its decision making process. This study investigates the impact of knowledge transfer on the achieved performances from several experiments.

The outline of the study is as follows: Section 2 introduces the PSO and AEPSO algorithms. The details about the simulated uncertainty are discussed in Section 3 and CAEPSO is presented in section 4. The experimental setup and the achieved results are presented in Section 5. Section 6 presents the conclusion.

2 PARTICLE SWARM OPTIMIZATION (PSO)

2.1 Basic PSO

Basic PSO is an evolutionary approach introduced by Kennedy and Eberhart in 1995 and it is inspired from animal social behaviors. PSO generates a population of possible solutions called *particles* (denoted by X) and evolve them toward optimum by iteratively readjusting particles' velocity (denoted by V). PSO takes advantage from cooperation among particles resulted from sharing their best findings with each other for achieving the optimum. The cooperation among particles in the swarm is facilitated through the use of social and cognitive components in the basic PSO velocity equation as shown in equation 1.

$$
\begin{aligned}
V_{i,j}(t) &= wV_{i,j}(t-1) + C_{i,j} + S_{i,j} \\
C_{i,j} &= c_1 r_{1,j} \times (p_{i,j}(t-1) - x_{i,j}(t-1)) \\
S_{i,j} &= c_2 r_{2,j} \times (g_{i,j}(t-1) - x_{i,j}(t-1))
\end{aligned} \tag{1}
$$

In the equation, C and S represent social and cognitive components. In the equation, i and j represent particle's index and particle's dimension in the search space respectively. t is the iteration number. r_1 and r_2 are random values between 0 and 1, w is the inertia weight. Linearly Decreasing the inertia weight (LDIW) formulated in equation 2 is one of the typical approaches used for adjusting w in basic PSO in which w_1 and w_2 the initial and final inertia weight, and *maxiter* is the maximum number of iterations.

$$
w = (w_1 - w_2) \times \frac{(maxiter - t)}{maxiter} + w_2 \tag{2}
$$

In equation 1, c_1 and c_2 are the acceleration coefficients used to control the impact of social and cognitive components on the readjustment of the particles' velocity. The cognitive component attracts the particles in the swarm toward their best personal findings (p). The social component attracts the particles in the swarm toward the global best findings (g). In PSO, personal and global best are updated using following equations:

$$
P_i(t) = \begin{cases} P_i(t-1) & if \quad f(x_i(t)) \geq f(P_i(t-1)) \\ x_i(t) & otherwise \end{cases} \tag{3}
$$

$$
g(t) = argmin\{f(P_1(t)), f(P_2(t)), ..., f(P_s(t))\} \tag{4}
$$

f represent the fitness (evaluator) function. The new position of each particle can be computed by following equation:

$$
x_{i,j}(t) = x_{i,j}(t-1) + V_{i,j}(t) \tag{5}
$$

A detail discussion about PSO, its advantages and shortcomings is presented in (Atyabi and Samadzadegan, 2011).

2.2 Area Extended PSO

This enhanced version of PSO is introduced aiming to solve basic PSO problems in robotic domains. The idea is based on using advanced versions of neighborhood topology and communication methodology with the aim of improving basic PSO performance in two dimensional multi-robot learning task in static, dynamic and noisy environments. In AEPSO, we solved fundamental problems[1] of basic PSO by adding some heuristics to it. Later on, the feasibility of the proposed modifications are examined in a simulated environment within several survivor rescuing scenarios. These heuristics are as follows:

a) To Handle Dynamic Velocity Adjustment. AEPSO takes advantage from a new velocity adjustment heuristic which tackles the premature convergence using equation 6.

$$
\overrightarrow{V}(t+1) = fittest \begin{cases} c_2 \times rand()(g(t) - x(t)) & & :1 \\ c_1 \times rand()(p(t) - x(t)) & & :2 \\ w \times V(t) & & :3 \\ 1+2 & HPSO & :4 \\ 1+3 & GPSO & :5 \\ 2+3 & GCPSO & :6 \\ 1+2+3 & Basic-PSO & :7 \end{cases} \tag{6}
$$

b) To Handle Direction and Fitness Criteria. AEPSO takes advantage from two heuristics known as Credit Assignment and Environment Reduction to addresses the cul-de-sac problem (Suranga, 2006).

[1]Fundamental problems of basic PSO are known as i)Lack of Dynamic Velocity Adjustment, ii)Premature Convergence, iii) Controlling Parameters and iv)Difficulties in Dynamic and Time Dependent environments(Peter, 1998; Mauris, 2002; Jakob and Jacques, 2002)

Environment Reduction Heuristic. The environment reduction heuristic is inspired from (Park et al., 2001; Yang and Gu, 2004) highlighting the advantage of separating a large learning space to several smaller spaces aiming to ease the exploration. In this heuristic, the large environment is divided into sub-virtual fixed areas with various credits. As in our simulations the environment is 500×500 pixels; each area is defined as a 20×20 pixels with square shapes resulting a matrix representation of 25×25 areas (625 areas overall). Each area contains 400 pixels with each pixel representing a possible location for obstacles, survivors, or robots. A credit associated to each area indicate the proportion of robots, survivors and obstacles positioned in that area. Only the likelihood information about the areas is provided to robots. An overall elimination time (iterations) for each area is also included in the areas credit to help robots to prioritize the observations of areas. This elimination time indicate the time left before all survivors in an area get eliminated. In here, exploration appears when a robot leaves its current area for another one and exploitation appears when a robot searches for survivors inside an area. Aiming for better balance between the exploration and the exploitation behaviors, robots only exploit areas that contain survivors (areas with positive credits). In order to increase robots' awareness, the likelihood information (areas' credits) of their neighboring areas are provided to them. These neighboring areas are divided to two layers of near and far neighboring areas as shown in fig. 1. In the simulation, robots set their direction according to the direction of the destination area and use maximum velocity whenever they want to leave an area for a new one.

1	2	3	4	5
16	1	2	3	6
15	8	current area	4	7
14	7	6	5	8
13	12	11	10	9

Figure 1: The first and the second neighboring areas of the current area.

Credit Assignment Heuristic. (Jim and Martinoli, 2007) argued that in some cases, as in macroscopic modeling of PSO, in a robotic problem, mathematical functions (benchmark functions) might not be appropriate as fitness evaluators since in such modeling, particles represent actual locations of robots in the environment. In AEPSO, a Punishment/Reward mechanism (inspired from reinforcement learning) is used instead. In here, robots would be rewarded positive credit whenever they find a survivor or whenever they locate themselves inside an area with positive credit. On other hand, robots would be punished by receiving negative credits if they do not achieve any reward after certain iterations or if they collide with obstacles.

c) To Handle Cooperation between Robots.

Communication Heuristic. The heuristic force robots to only communicate and share knowledge with those that are in their communication range which results in dynamic neighborhood topology and helps to create sub-swarms.

Help Request Heuristic. This heuristic provides cooperation between different sub-Swarms by allowing robots to request assistant and seek cooperation from other robots that are in their communication range whenever they require it. Robots that receive the request can either acknowledge the request or pass it through to others in their communicating range creating a chain of communication between robots that are far away from each other.

d) To Handle the Search Diversity.

Boundary Condition Heuristic. The heuristic solves the lack of diversity in basic PSO by forcing robots that get too close to the boundary of the environment to relocate themselves to somewhere in the middle of the environment by selecting an ad hoc direction and moving toward that direction for certain number of iterations.

AEPSO vs. PSO in a Simulated Survivor Rescuing Scenario.

In our simulation, we used variations of PSO as decision makers and movement controllers of autonomous robots. Fig. 2 (a and b) shows the results of basic-PSO in terms of trajectory traces in such simulation. The figs shows two different experiments with different initializations based on a survivor rescuing scenario in which team of 5 homogeneous robots are meant to find 15 survivors before they get eliminated. The environment is 500×500 meter and the simulated robots can detect objects (survivors or static obstacles) within a circle of 5 meter around them.

As the figs show, in basic PSO, the results are highly dependents to the initial locations of the instructor elements of the environment (robots, obstacles, survivals). Furthermore, due to the lack of balance between exploration and exploitation, robots were not able to cover a high percentage of the environment during their search while AEPSO was able to overcome the problem. The achieved mapping performance by AEPSO in addition to its ability to overcome random noise as it is demonstrated in (Atyabi et al., 2010) encouraged us to further evaluate the algorithm in simulated environments that are affected by highly complicated and more realistic type of noise.

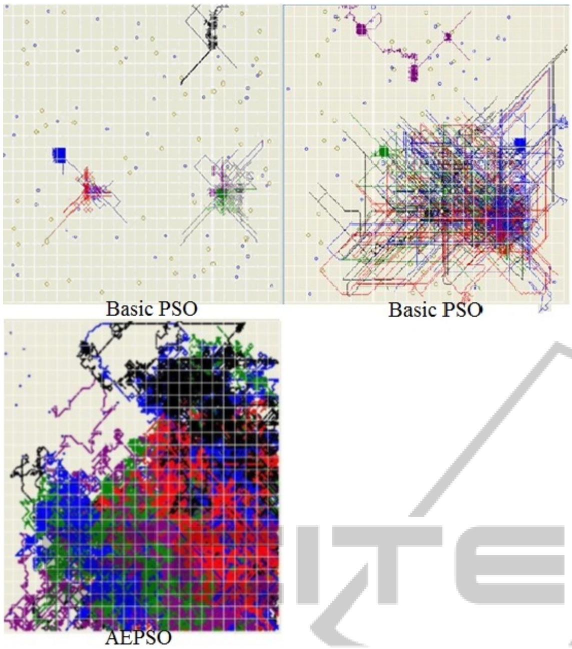

Figure 2: The trajectory traces of robots controlled by Basic PSO (two different executions) and AEPSO. different colors are used to represent different robots' trajectories. Blue and yellow dots represent survivors and obstacles respectively.

3 ILLUSION NOISE

The illusion noise presented in (Atyabi et al., 2010) is a combinatorial type of noise in which the noise value applied to the credit of each area in the environment is influenced from the neighboring (first layer of area neighborhood see fig 1) and far away areas (packs of neighborhood) and based on the ongoing activities in the environment the value of the noise in each area is changed over time (iteratively changing noise). In a simulated environment of 500×500 pixels divided to 625 different areas of 20×20 pixels resulting to a matrix of 25×25 cells, the application of the illusion effect results in a matrix representation of 25 equations needs to be predicted in each iteration in order to reveal true credit of each area for that iteration. Equation 7 shows the illusion credit of an area (this equation would be computed for each area in each of the iterations to provide the noisy credit of that area).

$$\begin{gathered} C_i(t+1) = C_i(t) + N(C_i(t)) + S(C_i(t)) \\ N(C_i(t)) = \sum_{j=1}^{8} (a \times actual\_credit(area_{i,j})) \\ S(C_i(t)) = \sum_{j=1}^{8} (b \times neighboring\_pack_{(i,j)})/8 \end{gathered} \tag{7}$$

C_i represents the corrupted value of an area affected by illusion noise in iteration $(t+1)$. a and b are constant parameters ($a = b = 0.125$) used to control the impact of j^{th} neighboring area and neighboring pack of area(i) (C_i). $N(C_i(t))$ and $S(C_i(t))$ represent the effect of other neighboring areas and neighboring packs on current area (C_i) respectively. $actual\_credit(area_{(i,j)})$ area is the uncorrupted credit of j^{th} neighboring area of $area_{(i)}$. $neighboring\_pack_{(i,j)}$ indicates the impact of neighboring packs of the pack which contains $area_{(i)}$. This impact is assumed as the average credit of areas who are the members of that pack.

Fig 3 illustrates an snapshot of the simulated environment which also demonstrate neighboring packs of an area.

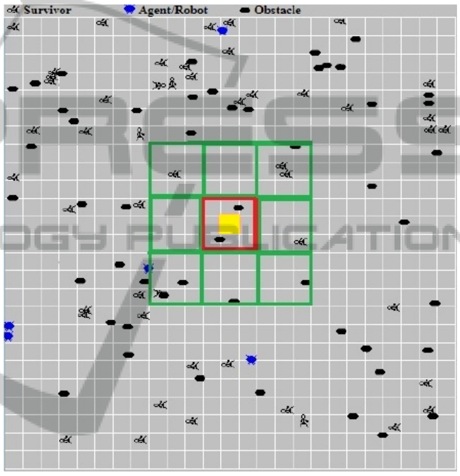

Figure 3: The environment during the initialization phase. Survivors, obstacles and agents are shown larger than the real experiment in which the size is equal to 1 pixel. White lines are used to virtually divide the environment to areas. Rectangles with green and red colors represent neighboring packs and neighboring areas of the current area respectively. The current area is illustrated with a rectangle filled with yellow color.

Fig 4 demonstrate the impact of the illusion effect on two simple pictures aiming to help with the understanding of the resulting complexity.

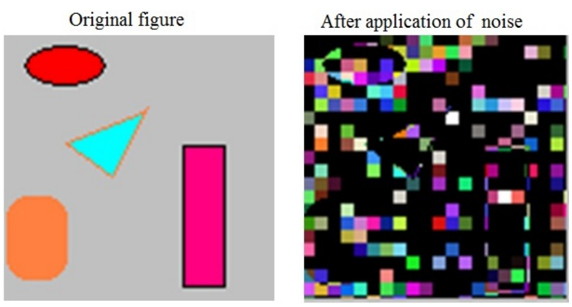

Figure 4: The impact of the illusion effect on a picture with 25×25 pixels. The left side picture is the original and the right side picture is after application of the illusion noise.

4 COOPERATIVE AREA EXTENSION OF PSO (CAEPSO)

Considering the complication that can be arised as the result of application of the illusion noise, the best possible way to solve the resulting huge puzzle is to identify the areas and neighboring packs that have the highest influence on others first. Identifying and mapping such areas results in eliminating their effects on other areas (by reducing their credits to zero). The decisions that robots make about the credit of each area are based on the elements such as Past knowledge, Current knowledge and perception, and some additional heuristics (Speculation mechanism).

4.1 CAEPSO's Additional Heuristics

In CAEPSO, two additional heuristics are suggested to tackle the complexities raised by the application of the illusion noise. These heuristics are as follows:

- **Leave Force.** The heuristic decreases 10% of an area's credit in a robot's mask whenever the robot enters the area or spent certain number of iterations exploiting that area (i.e., the area's flag would be changed to self speculation and the credit would be reduced by 10% off). The heuristic guarantee that robots do not spend a long time in an area and do not get stuck in areas.
- **Speculation Mechanism.** The heuristic helps to provide a high level of noise resistance. Speculation mechanism is based on using a small memory as a mask of the entire environment (i.e., a matrix of 25×25 cells, each cell representing an area in the simulated environment). In the start of a simulation, the value of each cell in the matrix represent the corrupted credit of the associated area in the environment (corrupted with the illusion noise) and later, robots update their mask of the environment based on their own observations and knowledge gained from other robots through knowledge sharing and communication. The value of robots' masks would be changed by the following factors:
  - robots' and their neighbors' self-observations which reduce the referring cells' value to zero. This happens whenever a robot fully map an area and locate and rescue all survivors within that area.
  - robots' and neighbors'-speculation which decreases the referring cells' value of their mask to the measured value.

  As robots share their masks with each whenever they are in each other communication range, they assess each others expertness to clarify the reliability of the knowledge they are receiving. That is, less expert robots only share their own or the others self-observations (the information that they are sure of); more expert robots also share their speculations about the areas' true credit. The robots degree of expertness are assessed based on factors such as the number of cells in their masks marked as self-observations and proportion of their rewards and punishments.

Following steps are taken when robots are within each other communication range (Knowledge sharing mechanism):

- Start of knowledge sharing
- Evaluate the expertness level of both robots (a and b)
  - The non-expert robot (a) only shares cells of its mask that are flagged as self and neighbor observation.
  - The Expert robot (b) shares cells of its mask that are flagged as self-observation, self-speculation, neighbor-observation, and neighbor-speculation.
- End of knowledge sharing

In an environment affected by illusion, each agent should choose an area for observation (exploitation) and their decisions can have significant effects on their own and group's performance (e.g., if they choose the best area (the area with the highest effect on others), they can help to reduce a high percentage of noise in the other areas and therefore, they can help to increase the group performance).

Pseudo-code of taken steps when robots are controlled by CAEPSO is presented in algorithm 1.

Algorithm 1: Pseudecode for controlling a robot with CAEPSO.

Step 1: Begin
Step 2: Initialization the robot is randomly located in the environment.
Step 3: use CAEPSO algorithm to update the robot's location.
Step 4: Checking the surrounding areas' credits, if it is changed due to robot's action then Speculation Mechanism is used to update the credits of surrounding areas in robot's memory.
Step 5: If all survivors are not located yet then go back to step 3.
Step 6: If the maximum number of iterations is not reached yet then go back to step 3.
Step 7: End

4.2 Past-knowledge

The major differences between AEPSO and CAEPSO are in the use of past knowledge provided by AEPSO during the training phase and two additional heuristics (Speculation and Leave-Force heuristics) in CAEPSO. A detailed discussion and description of AEPSO and CAEPSO can be found in (Atyabi, 2009; Atyabi et al., 2010).

Incorporating knowledge gained from multiple training sessions to increase the overall performance of swarm of robots proved to be advantageous in (Majid et al., 2001; Tangamchit et al., 2003). The training phase in robotic swarm can be designed to be either with individual or team of robots. Using a single robot for training is problematic given that the resulting information do not compensate the changes in the environment caused by other members of the swarm in the testing phase. On other hand, when a swarm of robots are used in the training phase, assessing the impact of the made decisions by individuals on the overall achieved performance is challenging if not impossible. In this study, a swarm of robots controlled by PSO are used in the training phase with an environment that is under the influence of illusion noise. Twenty runs of the experiment with random initial locations for robots, survivors and obstacles are executed and the gained knowledge is passed to the swarm of robots controlled by PSO in the testing phase (different initializations is used in the testing phase) [2]. The most important decision that robots make in such noisy environment is which area to exploit first. If areas with highest noise impact on others are chosen first, high percentages of the noise would be reduced from the environment. Considering the afore mentioned factor the past knowledge gained from the training phase is designed to reflect the potential of the made decisions in terms of the chosen directions (neighboring areas) with the individuals and the team of robots. Table 1 represent a template used for passing the gained overall knowledge from past training.

Table 1: The mask used to represent robots overall training phase knowledge (past knowledge).

a / b / c 1	a / b / c 2	a / b / c 3
a / b / c 8	Current Area	a / b / c 4
a / b / c 7	a / b / c 6	a / b / c 5

In the table, areas are denoted by numbers 1 to 8 with *a, b* and *c* representing the times that the area was the best area to be chosen, the times that the area was chosen as a positive credit area (potential direction), and the times that the chosen area was in fact the best area to be chosen respectively. Here, past knowledge refers to the overall knowledge gathered during the training phases from various trials/executions. In the testing phase, agents may be experiencing the same or new random initializations. Such a knowledge help agents to have overall information about their previous training and the quality of their previous decisions. The pseudo-code of CAEPSO is presented in algorithm 2.

Algorithm 2: CAEPSO Pseudecode.

```
Initialization: Randomly initialize the robots' locations in the environment.
Robots' masks' values= areas' credits affected by illusion
while (maximum number of iterations is reached or all survivors are found)
    do
        for (each robot in the swarm) do
            if (current area's credit = 0) then
                chose a new area for observation using environment
                reduction heuristic, mask, and past knowledge
            end
            else if (behavior = exploitation & performance is low) then
                behavior = exploration using leave force heuristic
            end
            else if (suspend factor or binary conditions are true) then
                use boundary condition, and credit assignment
                heuristics
            end
            else
                Update velocity using equation 6 and 2
                Update location using equation 5
                Evaluate new location using credit assignment
                heuristic
                Update mask using speculation mechanism
            end
        end
            Updating the personal_best using equation 3
            Communicate with robots located in the communication range
            Updating the global_best using equation 4
            Update robot's mask using speculation mechanism
    end
```

5 EXPERIMENTS AND RESULTS

A simulated environment with 500×500 pixels dimension, each pixel represent 1m, is used. The environment is polluted with illusion effect. Robots can only see within 5 pixels of their surroundings. A team of 5 robots are used for mapping the environment and locating the survivors. 15 and 50 static survivors and obstacles are randomly located in the environment. The maximum number of iterations is set to 20,000 while the elimination time for each survivor is set as a random value between 5,000 and 20,000. Robots task is to locate as many survivors possible before they get eliminated. The experiments are designed in two

[2]The difference between the initial locations used in the training and testing phases helps to better reflect real world situations in which the world is continuously changing.

phase. In the first phase, swarm of 5 robots controlled with AEPSO are randomly located in the environment with the task of finding the survivors. The decisions made by robots during 20 random runs of the experiment are evaluated and aggregated in mask and past to robots in the second phase (see an example of the mask in table 1). The experimental design and configuration in the second phase is similar to the first phase with the exception of using the past knowledge and CAEPSO as the controller of the robots[3]. Four sets of scenarios are designed in two phases of training and testing to address homogeneity and heterogeneity. The scenarios also investigate the potential of the transferred knowledge from the training phase when similar and new initializations are used in the testing phase. A detailed description of the experiments can be found in (Atyabi et al., 2010). In here, we are only interested in the impact of knowledge transfer on overall achievements across all scenarios discussed in (Atyabi, 2009; Atyabi et al., 2010).

The results in (Atyabi et al., 2010) showed transcendent improvement in terms of learned knowledge and robots' movement between training and testing phases. CAEPSO rescued 99% and 95% of the survivors during the testing phase with homogeneous and heterogeneous scenarios while robots controlled by AEPSO in the testing phase were only able to rescue 50% and 45% of the survivors. The past-knowledge provided to robots during the testing phase helped them to locate survivors in the testing phase regardless the differences between training and testing initialization. That suggest that CAEPSO is reliable in environments that have no direct past knowledge about it. The differences between the number of the eliminated survivors during the training and testing phases indicate CAEPSO's capability on overcoming the illusion effect and locating survivors in expected times.

Fig. 5[4] illustrate trajectory traces of 5 robots controlled by AEPSO in the first phase. The comparison between the results demonstrated in Figs 5 and 2 indicate inability of AEPSO to overcome the illusion effect evidenced by low portion of mapped environment. The aggregated results of the made choices by robots controlled by AEPSO during the first phase depicted in table 1 further indicate lost of overall performance due to inaccurate and inefficient choices made by robots in terms of which neighboring area to observe and map first. The wrong choices made by robots result in their inability to remove or reduce the effect of illusion noise from the environment.

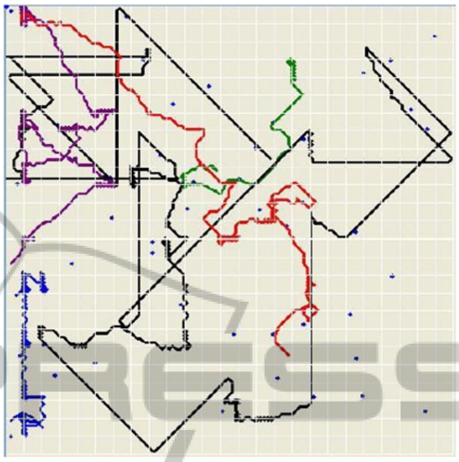

Figure 5: The training phase trajectory traces of a swarm of robots controlled by AEPSO in an environment corrupted by illusion. Different colors represent different robots. The figure is adapted from (Atyabi et al., 2010).

Table 2: Average of the aggregated knowledge from training phase of several experiments presented in (Atyabi et al., 2010).

10926 / 1 / 0 1	3343 / 0 / 0 2	671 / 1 / 0 3
19048 / 0 / 0 8		1195 / 0 / 0 4
12265 / 1 / 0 7	3639 / 0 / 0 6	1776 / 2 / 0 5

Table 3: Average of the aggregated knowledge from testing phase of several experiments presented in (Atyabi et al., 2010).

707 / 349 / 4 1	705 / 2590 / 51 2	707 / 438 / 5 3
51862 / 3996 / 3653 8		700 / 3931 / 46 4
774 / 416 / 5 7	1862 / 3988 / 126 6	714 / 394 / 4 5

Fig. 6[5] illustrate trajectory traces of 5 robots controlled by CAEPSO in phase 2 with and without past knowledge. Fig 6 (a) represents trajectory traces of 5 robots when no past knowledge is passed to them.

[3]In all experiments LDIW is used with $w_1 = 0.2$ and $w_2 = 1$ and Fix Acceleration Coefficients of $c_1 = 0.5$ and $c_2 = 2.5$ are employed. Other parameter settings and swarm configurations are considered and discussed in (Atyabi, 2009) among which the chosen setting showed consistently better overall performance.

[4]The figure is reprinted from Applied Soft Computing, 10, Atyabi et al., Navigating a robotic swarm in an uncharted 2D landscape, 49-169, (2010), with permission from Elsevier

[5]The sub figure 6(b) is reprinted from Applied Soft Computing, 10, Atyabi et al., Navigating a robotic swarm in an uncharted 2D landscape, 49-169, (2010), with permission from Elsevier

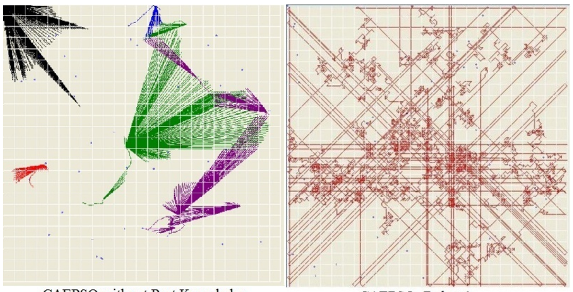

Figure 6: The testing phase trajectory traces of robots controlled by CAEPSO with (a) and without application of past knowledge. Different colors represent different robots. The survivor rescuing tasks are time dependent and the environment is corrupted by illusion.

The figure indicate occasional stagnation and robots inability to map a high percentage of the environment. In contrast, as evidenced in sub figure b, when past knowledge is presented to robots considerable percentage of the environment is mapped by each robot. This is also evident from the difference in the quality of the made decisions presented in tables 2 and 3. The combination of the presented results in tables 2 and 3 and fig 6 suggest the importance of knowledge transfer in environments polluted by combination of noises originating from different sources as in illusion noise.

6 CONCLUSIONS

This study discussed the impact of past knowledge on decisions made by a group of robots controlled by two variations of PSO called Area Extended PSO (AEPSO) and Cooperative AEPSO (CAEPSO). In order to evaluate such an impact a type of noise called Illusion effect is simulated. The illusion effect represent an iteratively changing noise that is the outcome of some combinations of noises originating from different sources located somewhere near or far away. The results of simulated experiments indicates the important role of past knowledge in compensating the illusion noise and making correct decisions by the simulated robots.

REFERENCES

Atyabi, A. (2009). Navigating agents in uncertain environments using particle swarm optimization. In *MSc Thesis*. Multimedia university of Malaysia.

Atyabi, A., Amnuaisuk, S. P., and Ho, C. K. (2010). Navigating a robotic swarm in an uncharted 2d landscape. In *Appl. Soft Comput*, pages 149–169. Elsevier.

Atyabi, A. and Samadzadegan, S. (2011). Particle swarm optimization: A survey. In *In Louis P. Walters, ed. Applications of Swarm Intelligence. Hauppauge, USA: Nova Publishers*, pages 167–178.

Grosan, C., Abraham, A., and Chis, M. (2006). Swarm intelligence in data mining. In *Springer, studies in computational intelligence (SCI)*, pages 34:1–20.

Jakob, S. V. and Jacques, R. (2002). Particle swarms extensions for improved local, multi-modal, and dynamic search in numerical optimization. In *MS.c Thesis, Dept. Computer Science, Univ Aarhus, Aarhus C, DenmarK*.

Jim, P. and Martinoli, A. (2007). Inspiring and modeling multi-robot search with particle swarm optimization. In *Proceeding of the 2007 IEEE Swarm Intelligence Symposium (SIS2007)*.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *IEEE Press Proceedings of the 1995 IEEE International Conference on Neural Networks*, pages 1942–1948.

Majid, N. A., Masoud, A., and Eiji, N. (2001). Cooperative q-learning: the knowledge sharing issue. In *Advanced Robotics*, pages 15(8): 815 – 832.

Mauris, C. (2002). The particle swarm - explosion, stability, and convergence in multidimensional complex space. In *IEEE Transaction on Evolutionary*.

Park, K. H., Kim, Y. J., and Kim, J. H. (2001). Modular q-learning based multi-agent cooperation for robot soccer. In *Robotics and Autonomous Systems*, pages 34:109–122.

Peter, J. A. (1998). Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In *Evolutionary Programming VII, Lecture Notes in Computer Science, Springer*.

Sousa, T., Neves, A., and Silva, A. (2004). Particle swarm based data mining algorithms for classification tasks. In *IEEE Parallel and nature-inspired computational paradigms and applications*, pages 30:767–783.

Suranga, H. (2006). Distributed online evolution for swarm robotics. In *Autonomous Agents and Multi Agent Systems*.

Tangamchit, P., Dolan, J. M., and Khosla, P. K. (2003). Crucial factors affecting cooperative multi-robot learning. In *Proceeding of the 2003 IEEE/RSJ Intl conference on intelligent Robots and Systems, Las Vegas, Nevada*, pages 2:2023–2028.

Yang, E. and Gu, D. (2004). Multi-agent reinforcement learning for multi-robot systems: A survey. In *Technical report, University of Essex Technical Report CSM-404, Department of Computer Science*.