J Intell Robot Syst (2010) 58:253–285
DOI 10.1007/s10846-009-9374-2

# Applying Area Extension PSO in Robotic Swarm

**Adham Atyabi · Somnuk Phon-Amnuaisuk · Chin Kuan Ho**

Received: 15 October 2008 / Accepted: 7 September 2009 / Published online: 13 October 2009


**Abstract** Particle Swarm optimization (*PSO*) is a search method inspired from the social behaviors of animals. *PSO* has been found to outperform other methods in various tasks. Area Extended *PSO* (*AEPSO*) is an enhanced version of *PSO* that achieves better performance by balancing its essential intelligent behaviours more intelligently. *AEPSO* incorporates knowledge with the aim of choosing proper behaviors in each situation. This study provides a comparison between the variations of *Basic PSO* and *AEPSO* aiming to address dynamic and time dependent constraint problems in simulated robotic search. The problem is set up in a multi-robot learning scenario. The scenario is based on the use of a team of simulated robots (hereafter referred to as agents) who participate in survivor rescuing missions. The experiments are classified into three simulations. At first, agents employ variations of *basic PSO* as their decision maker and movement controllers. The first simulation investigates the impacts of swarm size, parameter adjustment, and population density on agents' performance. Later, *AEPSO* is employed to improve the performance of the swarm in the same simulations. The final simulation investigates the feasibility of *AEPSO* in time-dependent, dynamic and uncertain environments. As shown by the results, *AEPSO* achieves an appreciable level of performance in dynamic, time-dependence and uncertain simulated environments and outperforms the variations of *basic PSO*, *Linear Search* and *Random Search* used in the simulations.

**Keywords** Area extension PSO · Dynamic · Particle swarm optimization · Robotic swarm · Time dependent · Uncertainty

A. Atyabi (✉) · S. Phon-Amnuaisuk · C. K. Ho
Faculty of Information Technology, Multimedia University, Cyberjaya, Malaysia
e-mail: adham.atyabi07@mmu.edu.my, aatyabi@acm.org

S. Phon-Amnuaisuk
e-mail: somnuk.amnuaisuk@mmu.edu.my

C. K. Ho
e-mail: ckho@mmu.edu.my

## 1 Introduction

Particle Swarm Optimization (*PSO*), introduced by Kennedy and Eberhart in 1995, has been inspired from animals' social behaviors which are illustrated by their social actions resulting in population survival [47]. *PSO* is a self-adaptive population based method in which, behavior of the swarm is iteratively generated from the combination of social and cognitive behaviors [4]. A swarm can be imagined as consisting of members called particles.[1] 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 (nearby neighbors in the flock). 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) provides several advantages for birds such as protection from predators and searching for food [23, 41].

In this study, an enhanced version of *PSO* called Particle Swarm Optimization with Area Extension (*AEPSO*) is introduced and its effectiveness in simulated environments affected by i) dynamicity, ii) time dependency and iii) uncertainty constraints is investigated. The problem is search optimization in hazard scenarios in which, a team of simulated robots participate in a survivor discovery mission in unpredictable environments.

The environment consists of goals (survivors with time limitations to be located), obstacles and agents that are randomly positioned during the initialization phase. The environment is $500 \times 500$ pixels and robots, obstacles and survivors size are considered as 1 pixel. The simulated robots' task is to locate survivors before they are eliminated (hereafter refer as elimination).[2] Furthermore, robots should evade collision with obstacles. Robots have no knowledge about the exact location of the survivors and obstacles. Robots (hereafter referred as agents) complete their observation task according to the acts and movements that *AEPSO* method provides for them in each iterations.

To measure the effectiveness of *AEPSO*, various simulations are suggested. In the first simulation, variations of *Basic PSO* are examined in static environments in which obstacles and survivors locations are fixed from the initialization. In this simulation, the impact of various swarm sizes and parameter adjustments on *basic PSO* are investigated.

In the second simulation, two environments are simulated (Static and Dynamic environments). In the static environment, survivors are randomly placed in different positions and they are fixed after the initialization while in the dynamic environment, the entire status of the environment changes iteratively due to survivors movement ability. In the third simulation, tasks' time dependency simulated in static and dynamic environments. Time dependency is simulated by defining various elimination times for survivors. In this simulation, agents have limitation in terms of communication range (this will effect knowledge sharing issue) and environmental perception (i.e., agents perception are affected by uncertainty).

[1] Also considered as: agents, observers or robots in [12, 13, 24, 54, 58].

[2] To improve the overall performance, simulated robots should give priority to survivors in a way that survivors with short elimination time be rescued first.

This paper is organized as follows. In Section 2, a brief description of *basic PSO* is presented. In Section 3, *PSO*'s enhancements and related works are discussed. Section 6 is dedicated to an enhanced version of *PSO* called Area Extension *PSO* (*AEPSO*). Simulations and empirical setups are presented in Section 7. Test results are discussed in Section 8. Discussion and conclusion are presented in Sections 9 and 10 respectively.

## 2 Background

### 2.1 Particle Swarm Optimization

In *basic PSO* [4, 5, 7, 8, 38], the actions of particles are based on their position in the search space denoted by $x_{i,j}$ and a velocity component in the n-dimensional search space denoted by $V_{i,j}$, where $i$ represents the particle's index and $j$ is the dimension in the search space. Particles fly through the virtual space and during that; they are attracted to positions (e.g., local and global-best positions) in the search space that yield the best results [1, 6, 52]. These positions in the search space are solutions and the global optimum is the global best solution (position) achieved by the entire swarm. In *basic PSO*, particles are affected from different neighborhood topologies (i.e., *local and global neighborhood*). The local neighbourhood,[3] is the neighborhood of a group of subset particles set as neighbors in a predefined way (particles with closest indices) [57]. The global neighbourhood[4] is a neighborhood containing all of swarm population [54]. In several studies, local neighborhood showed fast convergence toward optimum (local or global optimum) while global neighborhood guaranteed convergence toward global optimum even if it is slow [2, 48]. In *basic PSO*, a particle's memory contains local and global-best positions. The local-best position ($p_{i,j}$) is the position in which they achieved via their highest performance (personal best solution). Global-best position ($g$) is the best overall position of the neighbors. The velocity equation of the *basic PSO* contains three components: last velocity, cognitive and social components (with consideration to neighborhood topology) as shown in Eq. 1. The cognitive component (denoted by $C$), exploits the best local position. The social component (denoted by $S$) explores the global-best position based on neighborhood topology [16].

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

where, $r_{1,j}$ and $r_{2,j}$ are different random values in the range between 0 and 1 following the uniform distribution. $c_1$ and $c_2$ are known as acceleration coefficients. These two parameters control the effectiveness of social and cognitive components during the

[3] Also known as *local best*, *lbest*, and *ring topology* by literature in [2, 4, 57].

[4] Also known as *global best*, *gbest*, and *star topology* by literature in [2, 54].

solution finding process. The new position of each particle can be computed through the following equation:

$$x_{i,j}(t) = x_{i,j}(t-1) + V_{i,j}(t) \tag{2}$$

The velocity components of the particle ($V_{i,j}$) are limited to maximum and minimum allowable $V_{max}$ and $V_{min}$, as follows [41]:

$$V_{i,j} = \begin{cases} V_{min} & \text{if} \quad V_{i,j} \le V_{min} \\ V_{max} & \text{if} \quad V_{i,j} \ge V_{max} \\ V_{i,j} & \text{otherwise} \end{cases} \tag{3}$$

The value of $V_{max}$ is defined as one half of the total search range. The term inertia weight ($w$) in Eq. 1 is decreased linearly with time as suggested in [4, 8]:

$$w = (w_1 - w_2) \times \frac{(maxiter - t)}{maxiter} + w_2 \tag{4}$$

where, $w$ is inertia weight, $w_1$ and $w_2$ are the initial and final inertia weights, respectively. $t$ is the current iteration and *maxiter* is the termination iteration. Termination are due to achieving optimum or reaching to the maximum iteration (termination iteration). The inertia weight controls the effectiveness of previous velocity on the solution finding task (i.e., large and small values of inertia weight favors exploration and exploitation respectively). Thus, high inertia weight results in exploring the search space by avoiding local minima, while decreasing the inertia weight results in exploiting the search space and converging to the optimal solution.[5]

Following the velocity and position updates, the local and global-best positions at time $t$ are updated as follows [3, 18]:

$$P_i(t) = \begin{cases} P_i(t-1) & if \quad f(x_i(t)) \ge f(P_i(t-1)) \\ x_i(t) & otherwise \end{cases} \tag{5}$$

$$g(t) = argmin\{f(P_1(t)),\ f(P_2(t)),\ ...,\ f(P_s(t))\} \tag{6}$$

$f$ represents the fitness (evaluation) function.

Due to *PSO*'s potential in solving complex problems, it has been widely used in various scenarios and domains. Qin et al., in [43], introduced easier implementation and fewer amount of parameters as main advantages of *PSO* PSO compared to Genetic Algorithm *GA*. In many studies, *PSO* has been shown to perform as well as or better than *GA* in several instances. Eberhart and Kennedy found *PSO* perform on par with *GA* on the Schaffer f6 function [47]. In a study by Kennedy and Spears, a version of *PSO* outperformed *GA* in a factorial time-series experiment [49]. Furthermore, Fourie showed that *PSO* appears to outperform *GA* in optimizing several standard size shape design problems [50]. In works by Pugh, Zhang, and Martinoli, a local neighborhood version of *PSO* outperformed *GA* in a multi-robot learning scenario with homogeneous and heterogeneous robots [53–55].

Although *PSO* out-performed *GA* and other evolutionary algorithms in some problem solving tasks, it still suffers from some weaknesses which makes it brittle in

[5]The best experimental results were obtained by initializing the algorithm with a high inertia and linearly decreasing the value during the iterations [4].

some domains. The weaknesses of *PSO* and suggested modifications are presented in the following section.

## 3 PSO: Weaknesses and Enhancements

Although research results in different environments and problems show that *PSO* outperforms Genetic Algorithm (*GA*) in multi-robot learning and other group working-based problems [4, 8, 38] *PSO* has some weaknesses. These weaknesses are as follow:

1. Parameter control: Controlling parameters ( $c_1, c_2, w, w_1, w_2$) in *basic PSO* is a major issue (specifically in the velocity equation). These parameters have major parts/roles in controlling the effectiveness of social and cognitive components in finding optimal solutions. The following methods were used for controlling parameter values in various problems: *Time Varying Inertia Weigh (TVIW)* [4, 8, 38], *Linear Decreasing Inertia Weight (LDIW)* [38], *Time Varying Acceleration Coefficients (TVAC)* [4, 8, 38], *Threshold Model* [4], *Random* or even *Constant/Fix values (RANDIW, FAC)* [8]. Although the Threshold and *TVIW* models achieved the best performance compare to others, they are still not effective for dynamic systems. Furthermore, *TVAC* achieved better performances in multi-modal functions in contrast with *RANDIW* which can only be effective in unimodal functions. *FAC* and *RANDIW* methods results were poor in most of the problem solving domains [8].
2. Premature convergence: This problem mostly appears when one or more particles reach a local optimum and attract other particles to that point causing the swarm to converge to the location without any hope to achieve the global optimum. In this situation, particles slowly stagnate in the location due to the fact that other solutions around the local optimum have less fitness compared with the local optimum. It might also appear in situations in which particles flicker around global optimum and slow down somewhere around and near it (close clustering problem).
3. Lack of Dynamic velocity adjustment results the inability to hill-climb solution (e.g., premature convergence and lack of diversity). As Vesterstrom and Riget [4] mentioned, particles will flicker around aimlessly when they are settled near an optimum. It is due to the fact that the velocity vector never dynamically adjusts or even if it does, it would be so slow which will causes the performance to flatten out drastically near the optimum-although it is possible to solve this problem with linearly decreasing inertia weight (*LDIW*) method, the solution would be highly problem-dependent.
4. Difficulties on dynamic and time-dependent domains: *PSO* may not be reliable in domains with certain time constraints for solution finding tasks or in dynamic domains were the world (situation and states) would be changed in each period of time.

Earlier, literature has suggested various solutions and improvements to solve *basic PSO*'s problems. Some of these enhancements are listed as follow:

1. Improvements in terms of parameter adjustments. As it was mentioned, various parameter adjustments have been suggested to solve the *basic PSO* problems.

These adjustments are known as linearly decreasing inertia weight (*LDIW*), time varying inertia weight (*TVIW*), random inertia weight (*RANDIW*), fix inertia weight (*FIW*), time varying acceleration coefficient (*TVAC*), random acceleration coefficients (*RANDAC*), and fix acceleration coefficients (*FAC*). Furthermore, much literature has proposed to use new parameters to cope the premature convergence and search diversity problem [2, 4, 6, 18, 52].

2. Improvements in terms of velocity equation essential components. Literatures argued that in some problems, the algorithm would perform better if one ignores one of the essential components of velocity equation or even add a new component to them [8, 9, 38, 41, 51, 58].
3. Improvements in terms of neighborhood topology. As Kennedy and Mendes discussed in [2], in *basic PSO*, particles are affected by different neighborhood topologies (i.e., *local and global neighborhood*[6]). The local neighbourhood[7] is a neighborhood of a group of subset particles set as neighbors in a predefined way (particles with closest indices) [2, 4, 57]. The global neighbourhood[8] is a neighborhood containing all of the swarm population [2, 54]. In literature, local neighborhood showed fast convergence towards optimum (local or global optimum) while global neighborhood guaranteed convergence toward global optimum even if it is slow [2]. Kennedy in [2] discussed that it would be useful if both topologies used at the same time in a way that favors global optimum and fast convergence.
4. Mutation. Many studies suggested that it is possible to use a mutation factor whenever particles converged toward an objective. Such a factor helps them to explore other objectives or prevent premature convergence. It is also useful to solve the lack of search diversity [4].
5. Re-initialization. As literature suggested, by re-initializing particles whenever an objective has been found, it is quite likely to cope with the problem.
6. Clearing memory. Whenever an objective has been found, clearing the memory (personal best and global best) helps particles to find other objectives and it is as useful as re-initializing the particles. Such a factor helps them to explore for other objectives or prevent being trapped in local optimums [4].
7. Using Sub-Swarms. By introducing sub-swarms in a way that each sub-swarm's task is to optimize a specific objective, the *PSO* can handle the multi-objective problems.
8. Niching *PSO*. This technique is known as one of the best solutions for the problem in which sub swarms exists but not pre-fixed [14]. This will helps to improve the overall performance in multi-modal domains [15].

[6]Other topologies used by literatures are known as *Von Neumann, Star, Cluster, Pyramid,* and *Wheel* [2, 4, 48, 54].

[7]Also known as *local best*, *Circle*, *lbest*, and *ring topology* by literature [2, 4, 57].

[8]Also known as *global best*, *Star* and *gbest* by literature [2, 48, 54].

Quite a few articles have addressed improving the *PSO* model. However, conceptually, these attempts have not differed much from the *basic PSO* since the main investigation is on the updating of the velocity parameters to achieve better and faster convergence [30]. Although many attempts have been made to solve premature convergence or other major problems of *basic PSO* by adding new parameters and considering the diversity factor, most of them could not be introduced as a modified version which could achieve the level of real-world and dynamic environments. Furthermore, most of these methods have problems in the domain generality issue due to their domain dependency. Although *basic PSO*'s weaknesses provides difficulties in terms of algorithm adaptation in needed domain, *PSO*'s advantages such as less amount of parameters, less amount of computation, and algorithm simplicity motivated researchers to use it in various problems such as robotic swarm.

## 4 Robotic Swarm

Robotic Swarm refers to a population of logical or physical robots. The effectiveness of participation of such a swarm in complex problem solving tasks is due to the robots collaboration and cooperation abilities. These types of robots are widely used in military-based applications as bomb or threat detectors [46], moving products in big warehouses [40, 45], and search and rescue teams [45]. Research efforts have focused on methods based on the use of robots in hazard scenarios in which, central control is weak or even impossible (due to large distance, lack of communication, lack of information and so on). In such scenarios, the use of single intelligent robots is costly due to its time consuming nature, level of needed intelligence, and level of physical structure.

In [39], Bogatyreva and Shillerov suggested a hybrid method using hierarchical and stochastic approaches in a distributed swarm robotic system. In their study, the issue was to prevent chaos and perturbation in a path planning scenario. The study was done on logical robots (simulation-based application). Bogatyreva and Shillerov also suggested the possibility of adopting their proposed algorithm to a real world problem by using a modular system of basic robots that are able to build temporary structures (bridges, shelters) by self assembling with respect to environmental constriction as in [45]. The principle idea behind robotic swarm is to use local behavior of each robot to collectively and cooperatively solve a problem. As Liu and Passino [41] suggested, robotic swarm can be seen as a part of distributed artificial intelligence. This is due to coordination and cooperation of aggregated cohesiveness agents who make decisions. Hence, robotic swarms have been used in studies based on cooperative learning [26], problem solving [26, 44], and path planning [42, 43].

In [40], Werfel and Yaneer studied a simulation-based robotic swarm scenario based on goal locating. In this context, the goal refers to a building block. After locating the goals, robots should push them to another place for assembling. This is an example of simple, identical, autonomous, decentralized robots participating in structure building scenario. The idea was to implement a single robot with simple

skills that has the ability to build simple structures with low level of fault tolerance. Werfel and Yaneer suggested the expansion of the model to a multi-agent based system in which more than one robot participate in the structure building task. For such an aim, they introduced a lock factor on blocks to prevent the violation of geometric constraints in which more than one block is nominated to be attached in a similar place of the structure.

In [45], Mondada et al., used a team of simulated robots in a problem based on exploration, navigation and transportation of heavy objects on rough terrain. In their approach, robots are able to connect to each other and change their shape. Such a connection will provide a chain of robots which helps to cope with the problem of climbing obstacles or surf passing holes. The authors used a distributed adaptive control architecture inspired from ACO.

Some researchers suggested two models for implementing robot controllers in robotic swarm. These models are known as i) Macroscopic and ii) Microscopic models. In macroscopic model, the robotic swarm would be modeled as a whole, while in microscopic model; each robot would be modeled separately.[9] The concepts of macroscopic and microscopic modeling have been used by many other researchers in various robotic swarm studies [34, 59–62]. Recently, Pug and Martoinoli [56] and Li et al., [63] used macroscopic and microscopic modeling in their studies. In [56], Pugh and Martinoli used macroscopic modeling in a team of robots that participate in a multi agent search. Li et al. in [63] used microscopic modeling in a robotic swarm. In [63], agents are represented as separate Probabilistic Finite-State Machines (PFSM). The authors suggested that by following microscopic modeling, it is possible to study all levels of swarm diversity. Furthermore, they argued that this type of modeling can address both homogeneity and heterogeneity issues in agents (due to its ability to represent agents with different PFSMs).

In this study, we followed the macroscopic modeling idea by introducing a new modified version of *PSO* called Particle Swarm Optimization with Area Extension (*AEPSO*). The effectiveness of suggested method is investigated in i) dynamic, ii) time dependent and iii) uncertain environments. In our simulations, the environment consists of goals (survivors with various elimination times), obstacles and agents with a randomly position in initialization phase. The environment is $500 \times 500$ pixels and agents should rescue survivors as fast as possible while preventing collision with obstacles. Simulated agents participate in survivor rescuing missions considering *AEPSO*'s instructions. In the study, the effectiveness of suggested methods are investigated with various simulations.

As it would be discussed, *AEPSO* has the potential to locate the desired goals faster. It is due to its ability to provide a balance between exploration and exploitation during the search. This balance is achieved by introducing and using heuristics. In the following sections, *AEPSO* and their additional heuristics are presented.

[9]To implement agent controller in a robotic swarm by *PSO*, macroscopic model refer to implementations in which each agent represent a particle of the swarm (therefore, the whole team represent the swarm) while microscopic model refer to implementation in which each agent represent a swarm by itself and particles of that swarm, represent the possible acts for that individual agents as the next move.

## 5 Problem Statement

To define the problem, the following *representations* are used:

| Term | Value |
|---|---|
| Agents : *(A)* | $\{a_1, ..., a_m\}, 5 \leq m \leq 20$ |
| Survivors : *(S)* | $\{s_1, ..., s_k\}, 1 \leq k \leq 50$ |
| Obstacles : *(O)* | $\{o_1, ..., o_n\}, 1 \leq n \leq 50$ |
| Environment : *(E)* | $\{(q_1, q_2) \in N \mid (0 \leq q_1 \leq 500), (0 \leq q_2 \leq 500)\}$ |
| *Agent* $a_i$ *characteristics* | |
| Location | $x_i^a \in E$ |
| Velocity | $v_i^a \in \{1, 2, 3, 4\}$ |
| Collision | $cl_i \in \{True, False\}$ |
| Evaluation | $f(a_i) \in \{Rewarded, Punished\}$ |
| Observation range | $r = 3$ |
| Communication range | $cr \in \{5, 125, 250, 500\}$ |
| *Survivor* $s_i$ *characteristics* | |
| Location | $x_i^s \in E$ |
| Velocity | $v_i^s \in [0, 3]$ |
| Rescue | $rs_i \in \{Rescued, \neg Rescued\}$ |
| Eliminate status | $el_i \in \{Eliminated, \neg Eliminated\}$ |
| Elimination time | $(3000 \leq t_i \leq 20000)$ |
| *Obstacle* $o_i$ *characteristic* | |
| Location | $x_i^o \in E$ |

Let $A = \{a_1, ..., a_m\}$ be a set of $m$ cooperative agents (robots) on a cooperative team. Agents move in an environment *(E)* represented as a *2D* landscape. Let $S = \{s_1, ..., s_k\}$ and $O = \{o_1, ..., o_n\}$ represent a set of survivors and obstacles respectively. Detailed characteristics of agents, survivors, and obstacles are given as follows:

1. Each $a_i$ in $A$ has number of characteristics represented as $< x_i^a, v_i^a, r, cr_i, cl_i, f(a_i) >$. $x_i^a$ and $v_i^a$ are location and velocity of $a_i$ in the $E$ respectively, $i$ is the agent's index and $j$ is the dimension in the search space. $r$ is agent's observation range. $cr$ is the communication range of the agent, $cl_i$ represents that either the agent $a_i$ collide with an obstacle in its previous location or not. $f(a_i)$ represents the result of evaluation of agent $a_i$'s location.
2. Each $s_i$ in $S$ has number of characteristics represented as $< x_i^s, v_i^s, t_i, rs_i, el_i >$. $x_i^s$ and $v_i^s$ are location and velocity of $s_i$ in the $E$ respectively. Survivors' movement provides dynamism in the $E$. Survivor $s_i$ would be either rescued or eliminated. These represent with $rs_i$ and $el_i$ respectively. The $t_i$ is the survivor's elimination iterations.
3. Each $o_i$ in $O$ has $< x_i^o >$ as its characteristic. $x_i^o$ represents location of $o_i$ in the $E$.

*Problem:* Given initial $x_i^a, x_i^s, x_i^o$ of $A$, $O$, and $S$ in $E$, rescue all survivors before they eliminate and avoid obstacle collision.

1. Assume that $\forall\, s_i \in S, \forall\, o_i \in O, \forall\, a_i \in A,\ x_i^a, x_i^s, x_i^o \in E$.
2. ***Communicate and share knowledge:*** $\forall\, a_i \in A,\ \forall\, a_j \in A,\ (d(a_i, a_j) \leq cr)$. $d(a_i, a_j)$ represents the distance of agent $a_i$ from agent $a_j$ in the $E$.

3. ***Rescue:*** $\forall\ s_i \in S,\ \forall\ a_i \in A,\ \ [(d(s_i, a_i) \leq r)\ \wedge\ (t_i \geq t)\ \wedge\ (rs_i = \neg Rescued)\ \wedge\ (el_i = \neg Eliminated)] => (rs_i = Rescued \wedge el_i = \neg Eliminated)$. $d(s_i, a_i)$ represents the distance of $s_i$ from $a_i$ in the $E$. $t$ represents the iteration and $0 \leq t \leq 20000$.
4. ***Eliminate:*** $\forall\ s_i \in S,\ [\ (t_i < t)\ \wedge\ (rs_i = \neg Rescued)\ \wedge\ (el_i = \neg Eliminated)\ ] => (rs_i = \neg Rescued\ \wedge\ el_i = Eliminated)$.
5. ***Collide:*** $\forall\ a_i \in A,\ \exists\ o_i \in O,\ (d(o_i, a_i) = 0) => (cl_i = True)$.

The basic terms that are used in the study are presented in Table 1.

**Table 1** Basic terms and definitions

| | |
|---|---|
| *Area* | Refers to a group of pixels and it is used to ease the search and reduce the environment dimensions. |
| *Area neighborhood* | Refers to a group of areas that are located near the current area. In our simulations, agents have the knowledge about the credit of areas that are located in the first and second layers of the area neighborhood. These layers contain eight and sixteen areas surrounding the current area as in Fig. 1 |
| *Agent* | Refers to an autonomous robot. |
| *Credit* | Refers to a value which represents agents' idea about the worth of an area in term of exploitation. In here, credit value represents the amount of survivors, obstacles, and agents inside an area. |
| *Diversity* | Refers to the proportion of agents' using exploration and exploitation behaviors. |
| *Elimination* | Refers to a situation in which survivors would die and be cleared from the environment. |
| *Mission* | Refers to a test in which agents search for survivors with the aim of locating them before they reach their elimination time (iteration). |
| *Neighboring areas* | Refers to a group of areas that share boundaries with the current area. In our simulations, neighboring areas contains areas known as first layer of area neighborhood. |
| *Neighborhood pack* | Refers to a group of eight areas surrounding a centering area. These eight areas are neighboring areas of the centering area. |
| *Noise* | Refers to random values added to areas' credits |
| *Positive credit area* | Refers to areas that contain un-rescued survivors. |
| *Rescue* | Survivors would be counted as rescued whenever they located by agents. the rescued and eliminated survivors would be cut out of the environment. |
| *Survivor* | Refers to goal with a certain living (rescuing) time limit. |
| *Simulation* | Refers to a group of tests in which a particular constraint is mimicked from real world. |
| *Termination criteria* | Refers to situations in which no survivors are remained in the environment. Either they are eliminated or rescued. |
| *Time-dependency* | Refers to simulations in which, survivors have specific living times (here iterations). Survivors would be death and cleared from the environment if agents do not locate them by the time. |
| *Uncertainty* | Refers to situations in which agents' environmental perceptions (here areas' credits) are not reliable. Uncertainty is simulated as random noise (a distributed uniform value dedicated to areas' credits iteratively). |
| *Velocity* | Refers to the step-size of agents and it is confined in maximum and minimum allowable values ($V_{max}$ and $V_{min}$). |

## 6 Particle Swarm Optimization with Area Extension (*AEPSO*)

This new enhanced version of *PSO* is introduced with the aim of addressing *basic PSO*'s problems in robotic domain. The idea is based on using advanced neighborhood topology (dynamic neighborhood topology) and communication methods with the aim of improving *basic PSO* performance in two dimensional multi-robot learning tasks in static, dynamic, time dependent and uncertain simulated environments.

In *AEPSO*, we attempt to address fundamental problems of *basic PSO* by adding these heuristics to it:

1. To handle dynamic velocity adjustment:
   - New velocity heuristic which address the premature convergence [4, 8, 38, 58].
2. To handle direction and fitness criteria:
   - Credit Assignment heuristic which address the cul de sacs problem [26, 37].
   - Environment Reduction heuristic.
3. To handle communication limitation in real-world robotic domains:
   - Different communications ranges condition which provides dynamic neighborhood and sub-swarms [2, 4, 14, 56].
   - Help Request Signal which provides cooperation between different sub-swarms [34].
4. To handle the search diversity:
   - Boundary Condition heuristic which address the lack of diversity in the *basic PSO* [4, 8, 58].

Pseudo-code of *AEPSO* is as follows:

*AEPSO's Pseudo-code*

```
parameters and locations initialization
While t ≤ Maxiter do
{
  For each a_i ∈ A
  {
    If (current area's credit = 0)
    Then choose a new area
                                   /* use environment reduction heuristic */
                                   /* Exploration Behavior */
    Else
      If ((agent's a_i behavior = exploitation) and (performance is low))
        Then behavior = exploration /* use leave force heuristic */
                                   /* Exploitation Behavior */
    Else
      If (Suspend factor or boundary condition are True)
        Then explore the environment
          /* use boundary condition and credit assignment heuristic */
                                   /* Exploration Behavior */
    Else
                                   /* Exploitation Behavior */
      {
        update velocity (v_i^a)                    /* use equation 7 */
        check the velocity limitation              /* use equation 3 */
        update location (x_i^a)                    /* use equation 2 */
        evaluate the agent a_i (f(a_i))  /* use credit assignment heuristic */
      }
    update personal_best                           /* use equation 5 */
    communicate with neighboring agents (share environment masks)
                  /* use communication heuristic and help request signal */
    update gloabal_best                            /* use equation 6 */
  }
}
```

In the pseudo-code, $t$ refers to the current iteration and *Maxiter* refers to the maximum allowable iteration (here 20000 iterations). Moreover, *current area's credit* refers to the amount of un-rescued survivors in an area. In the pseudo-code, /∗ and ∗/ are used to provide comments and distinguish them from the code. *Exploration* refers to choosing a new area and moving toward it by setting the direction and choosing $V_{max}$ as velocity. *Exploitation* refers to investigating locations inside an area (a positive-credit-area) using *AEPSO*'s velocity Eq. 7.

As the pseudo-code depicts, *AEPSO*'s algorithm replicates in each agent. In each iteration, agents take following steps. First, each agent checks its current area (the area it is location in). If the area contains no un-rescued un-eliminated survivor, the agent changes its behavior to exploration by setting its velocity to $V_{max}$ and updating its $x$ according to the new $V$ and the direction of the destine area. The destine area would be choose based on first and second layers of neighboring areas which indicates the credit of neighboring areas (see environment reduction heuristic). Later, agents check their performance and get suspended if the performance is lower than a predefined value. Agents get suspended by changing their behavior to exploration and setting suspend value (see credit assignment heuristic). Agents also get suspended if their location is so near to the boundary lines. As a result, agents use boundary condition heuristic's instructions. If none of previously mentioned conditions are true, agents update their velocity and location using Eqs. 2, 3, and 7. As the next step, agents evaluate their new location using credit assignment heuristic. Finally, agents communicate with each other based on communication heuristic's instruction. This helps agents to gain new knowledge and update global best. The following sections represent details of used heuristics in the *AEPSO*'s pseudo-code.

### 6.1 Dynamic Velocity Adjustment

In *AEPSO*, in each of the iterations, we used one of the three essential components of *basic PSO* (last velocity, cognitive, and social components) or one of their four extra combinations which could shift an agent to an area or position with the best credit (fittest position of the iteration for that agent) as shown in Eq. 7.

$$\overrightarrow{V}_i(t+1) = fittest \begin{cases} \phi_2(g(t) - x_i(t)) & & : 1 \\ \phi_1(p_i(t) - x_i(t)) & & : 2 \\ w \times \overrightarrow{V}_i(t) & & : 3 \\ 1+2 & HPSO & : 4 \\ 1+3 & GPSO & : 5 \\ 2+3 & GCPSO & : 6 \\ 1+2+3 & Basic-PSO & : 7 \end{cases} \tag{7}$$

$$\begin{aligned} where \quad \phi_1 &= c_1 \times rand() \\ \phi_2 &= c_2 \times rand() \end{aligned} \tag{8}$$

Parameters ( $w$, $\phi 1$, $\phi 2$, $c_1$, $c_2$, $p_i$, $g$) and strategies that have been applied to control them are discussed in previous sections. Equations 5 and 6 are used for new $p_i$ and $g$ in each of the iterations for each agent. Linearly Decreasing Inertia Weight (*LDIW*) is used as in Eq. 4. The new modified velocity equation provides extra control on effectiveness of different components of the velocity in a way to achieve

the best solution possible for each agent in each iterations. Although the new velocity equation needs the computation of seven quantities, it can provide dynamic velocity adjustment due to its ability to provide different behaviors in the swarm at the same time. In *AEPSO*, each agent represents a particle of the swarm (macroscopic modeling). As discussed in previous sections, in microscopic modeling of a robotic swarm, each agent represents a swarm by itself and a large amount of particles would be suggested as swarm-size for each agent/swarm (at least 20 particles have been used as swarm-size by literatures [34, 37]). The use of new velocity equation (Eq. 7) in such a model will cause a high amount of computation which can slow the process while by macroscopic modeling of the swarm, as in this study, the swarm-size would be equal to the amount of agents used in suggested simulations. Therefore, by imitating the macroscopic modeling, it is preferable to use the benefits of such a velocity equation due to the fact that the amount of computations are already reduced.

### 6.2 Environment Reduction Heuristic

As argued in some literature [64–66], it is possible to separate a large learning space to several small learning spaces with the aim of easing the exploration. The idea in environment reduction heuristic is to divide the environment to sub-virtual fixed areas with various credits. In this study, the environment is divided to $20\times20$ pixels with square shapes cells (areas). Since the environment is presumed as $500\times500$ pixels; a matrix of $25\times25$ areas (625 areas overall) is concluded in a way that each area contains 400 pixels.[10] The credit of each area represents the proportion of agents, survivors and obstacles positioned in the area. Agents have no knowledge about the exact location of objects (survivors and obstacles) inside areas. Agents have knowledge about overall elimination time (iterations) of each area. This can be used to give the priority to areas with lower elimination times.

In this study, exploration behaviors refers to situations in which an agent leaves its current area for another one. What's more, exploitation behavior refers to situations in which an agent searches for survivors inside an area. Since the environment reduction heuristic forces agents to only exploit in areas with positive credits[11] and explore in other situations, following such rule helps to provide more effective balance between exploration and exploitation behaviors. In *AEPSO*, agents use the credit of their surrounding areas for choosing areas for exploration. To do so, agents have knowledge about the credit of their first and second surrounding areas. These areas are referred as neighboring areas. These neighboring layers of areas contain eight and sixteen areas around current area[12] and are depicted in Fig. 1.

Due to the fact that agents give priority to areas for exploitation (based on areas' credits), they set their behavior to exploration and also set their heading directions based on the destine area and use maximum velocity ($V_{max}$). Agents change their behavior to exploitation whenever they reach to the destine area. In this study, two different Area Deserting Policies (*ADP*) are assumed (*ADP1* and *ADP2*). In

[10] Since the size of agents, obstacles and survivors are considered as one pixel, each pixel in an area represents a possible survivor location for agents.

[11] An area would be referred as a positive credit area whenever it contains un-rescued survivors.

[12] Current area is the area that the agent is posed inside it.

**Fig. 1** Different layers of area's neighborhood

| | | | | |
|---|---|---|---|---|
| 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 |

the *ADP1*, agents are not allowed to leave an area unless they rescue all of the remaining survivors inside that area. In the *ADP2*, agents are allowed to leave an area without completing survivors' rescuing tasks inside that area. The *ADP2* allows agents to avoid getting temporarily stuck and loops of movements and directions (e.g., especially in uncertain environments). It is due to their ability to change their status by leaving the area. It is still possible for agents to be re-attracted to the same area during the next iterations.

According to the algorithm, global-best and personal-best positions would be cleared from the memory of an agent whenever it reaches a position which is marked as global-best or personal-best position. Hence, whenever the global-best or personal-best positions are unknown (previously cleared), agents use a random location inside their current areas and the environment, respectively. It is necessary to consider that agents share their personal-best-position and personal-best-position during communications.

In *AEPSO*, agents move to each of the eight different areas around their current area (first layer of neighboring areas) with consideration to their credits and by changing their velocity to $V_{max}$ as shown in Fig. 2 each time they decide to leave their current areas. If none of these neighboring areas had a credit more than zero, they will choose their new area from the second neighboring areas.

In *PSO*, $V_{max}$ is defined as the maximum velocity value. As [56] discussed, in *PSO*, particles have no intrinsic limitation in terms of maximum velocity. It is necessary to be mentioned that even if there is a velocity limitation, it is common to use a high value for $V_{max}$. This allows particles of the *PSO* to fly in the virtual space toward the interest regions in a single step. However, by considering the fact that in macroscopic modeling of *PSO*, as in this study, the $V_{max}$ represents the step-size of the agents, it is impossible to assign high values to $V_{max}$. As it would be discussed in following sections, in our simulation, we defined $V_{max}$ as four pixels per iteration.

### 6.3 Communication Methodology

It is common to use pre-established connections following neighborhood topology to model communication between particles in *PSO*. Hence, particles' locations in the search space do not influence their communications [2]. In contrast, in real-world robotic applications, robots' communications influence from their location in the environment. In real-world robotic domains, robots' communication range have major role [55, 56]. This cause the existence of dynamic neighborhood topology. In dynamic

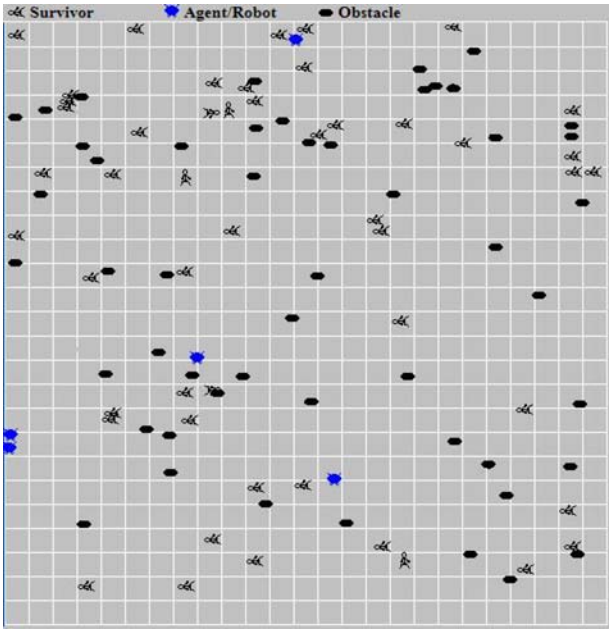



**Fig. 2** The environment during the initialization phase. Survivors, obstacles and agents are shown larger than the real experiment in which the size is equal to 1 pixel. *White lines* are used to virtually divide the environment to areas

communication methodology, neighborhood topology would be continuously change based on the distance of agents from each other. During communication, agents share *local-best-position (p), global-best-position (g)*. Even though memorizing previous acts and last-observed areas are common in robotic problems, this study emphasize on the use of small amount of memory (personal and global-best positions).[13]

### 6.4 Help Request Signal

Help request signal is considered as signal that agents send to each other whenever they found themselves in areas with low elimination time (near to death people).[14] In this study, agent's communication range influence signal sending issue in a way that agents can only send their help request signals to those who are in their communication range. Therefore, whenever an agent receives a help request signal, it will answer the request by either leaving its current location to help the requester or ignoring the signal and continue with its previous act. Agents that receive the request will also re-send it to others who are in their communication ranges. This results in a virtual chain of connection with which, the original requester would be connected others that are far away. Due to such a connection, agents are able to attract other agents that are far away from their location. This method can also handle cases in which agents use various communication ranges.

[13]Using short memory during the communication and knowledge sharing issues is one of the essential ideas of *basic PSO*.

[14]In this study, agents have knowledge about the amount of survivors, obstacles and other agents inside the area they are located in. Moreover, agents have knowledge about the overall elimination time of the survivors inside their area.

### 6.5 Credit Assignment

As Pugh and Martinoli argued in [56], in some problems (as in macroscopic modeling of *PSO* in a robotic problem), the use of mathematical functions (benchmark functions) as fitness evaluators might not be appropriated. In such modeling, particles of the swarm are referring to actual locations of agents in the environment. Therefor, in this study, three new terms called *Reward, Punishment,* and *Suspend* are used credit assignment heuristic. In this study, agents would be either rewarded or punished based on their status (locations) in the simulated environment. Hence, a reward.[15] would be dedicated to an agent either when it locate a survivor or when it locate itself inside an area with positive credit[16] Likewise, Punishment would be dedicated to a agent either when it could not achieve any reward after certain iterations or when it collide with obstacles.

In *AEPSO*, the suspend entity (suspend factor) is designed to force agent to leave their current area in situation in which they do not achieve performance for some iteration. In here, suspend factor in agents is simulated by changing the behavior to exploration and using $V_{max}$ as velocity and a new random direction (for certain duration). This entity would be used for agents that received high amounts of punishments. Punishment and reward mechanism motivates agents to avoid resting. It is also useful in situations in which agents stuck between obstacles.[17] Since credit assignment heuristic forces agents to locate themselves in a far distance from the trap zone (due to using the suspend factor) it is a promising solution for the stagnation problem. Therefore, *AEPSO* considers the following instructions to evaluate agents' achievements.

$$f(a_{(i)}) = \begin{cases} Reward <= [\exists s_i \in S, d(s_i, a_i) \leq r] \\ \qquad\qquad \vee [current\ area\ is\ a\ positive\_credit\_area] \\ Punish <= [\neg\exists s_i \in S, d(s_i, a_i) \leq r] \\ \qquad\qquad \wedge [current\ area\ is\ not\ a\ positive\_credit\_area] \\ Suspend <= [Punishement\ value\ is\ higher\ than\ maximum\ allowable] \end{cases}$$

$f$ represents the results of evaluation (fitness) of agent $a_i$. $S, s_i, a_i, d(s_i, a_i)$ and $r$ are introduced in problem statement (Section 5).

### 6.6 Boundary Conditions

In this study, the boundary conditions determine *AEPSO*'s policy whenever agents are exploiting around the environment boundaries and cross them.[18] As it is shown in numerous studies with *basic PSO*, it often happens that particles position themselves

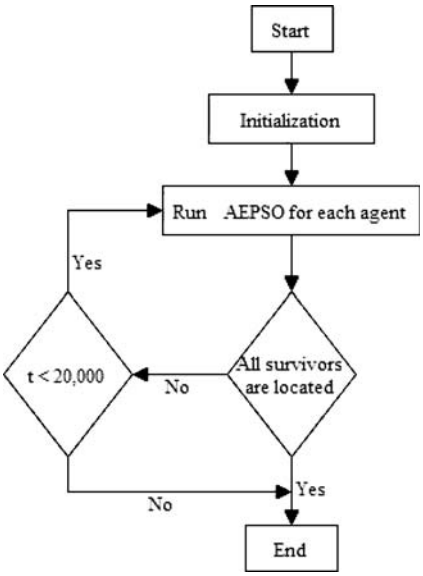
[15] In here, reward and punishment are considered as positive and negative units of credit.

[16] A positive-credit-area is assumed as an area with un-rescued survivors.

[17] In [37], it is mentioned that in robotic domain, robots are often stuck (trapped) between obstacles without any hope to be released. This problem causes robots to lose performance and it is referred as cul de sac or stagnation problem.

[18] Even though agents have knowledge about the location of boundaries, they would consider themselves crossing them whenever they see them in their observation ranges and pass them.

**Fig. 3** Algorithm's flowchart




in a location out of the search space ( by crossing the search boundaries) [4]. The evaluation of such particles causes negative effects on the overall performance of the swarm. It is due to the fact that these particles are referring to locations out of the space. Evaluation of such particles also motivates other particles to imitate same behavior and converge toward that location by leaving the search space. Vesterstorm and Riget in [4] suggested following strategies to handle such a problem:

- Relocating particles to locations inside the search space or somewhere on the boundary line.
- Ignoring particles miss-position and evaluate them as they are on the boundary lines.
- Re-initializing particles.[19]

In this study, the third solution (re-initializing particles) is used to handle boundary problem. Since it is not possible to relocate agents in robotic problem, re-initialization is simulated by forcing agents to follow a random direction for certain iteration using suspend factor. Therefore, agents will change their behavior to exploration whenever they crossed the boundary lines. Likewise, agents will choose ($V_{max}$) as their velocity. They will also choose a random direction and set their suspend factor true.

As it is mentioned in [22], this heuristic is useful when agents are flickering around the boundary lines while survivors are placed in a different part of the environment. The idea of forcing agents to relocate themselves somewhere in the middle of the environment shows a large influence on agents' performance.

The used steps in *AEPSO* are illustrated in flowcharts in Figs. 3 and 4. Figure 3 depicts the algorithm's flowchart. In the flowchart, in the initialization step, a random

[19]Other strategies such as *Warping* and *Reflecting* have also been examined in robotic by literature.

location would be dedicated to agents, obstacles and survivors. In addition, each survivor's elimination time would be set randomly with a value between 5000 to 20000 iterations. Figure 4 demonstrates the use of heuristics in the algorithm. In the flowchart, the value of suspend factor depict the iteration durations that agents are suspended.

## 7 Simulations and Empirical Setups

In this study, three different simulations are defined to measure the effectiveness of *AEPSO* in different realistic environments. These simulations represent various constraints which are common in real world applications. These constraints are

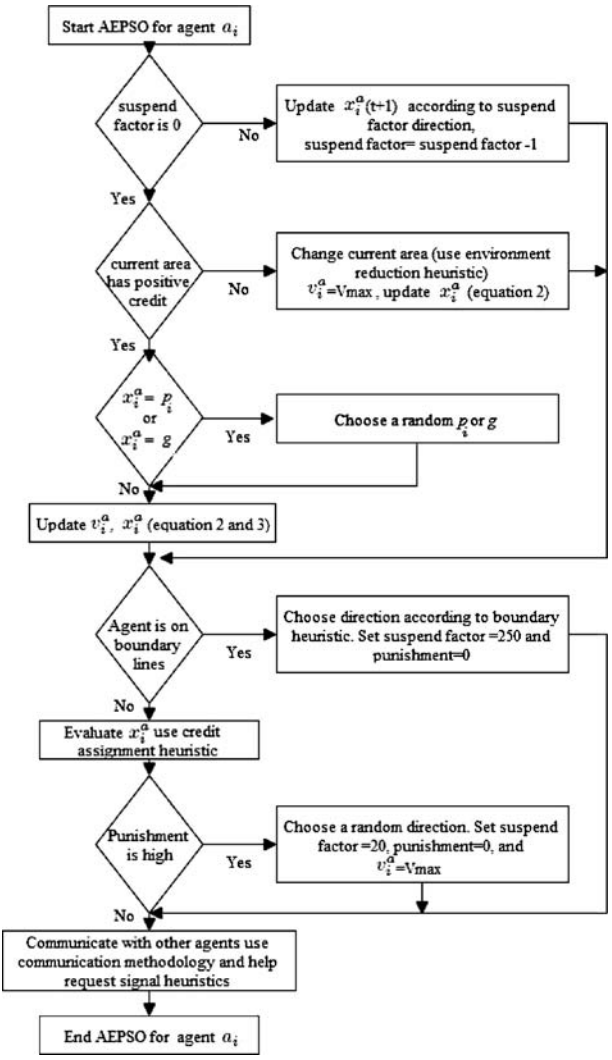



**Fig. 4** Details about *AEPSO*'s use in each agent

**Table 2** AEPSO's parameter setups in various simulations

| Parameters | Simulations | | | |
|---|---|---|---|---|
| | Simulation 2 | | Simulation 3 | |
| | Static | Dynamic | *TS* | *TD* |
| $V_{max}$ | 4 | 4 | 4 | 4 |
| $V_{min}$ | 1 | 1 | 1 | 1 |
| $c_1$ | 0.5 | 0.5 | 0.5 | 0.5 |
| $c_2$ | 2.5 | 2.5 | 2.5 | 2.5 |
| $w_1$ | 0.2 | 0.2 | 0.2 | 0.2 |
| $w_2$ | 1 | 1 | 1 | 1 |
| Observation-radius | 3 | 3 | 3 | 3 |
| Obstacles | 51 | 51 | 45 | 45 |
| Survivors | 51 | 51 | 15 | 15 |
| Agents | 5 | 5 | 5 | 5 |
| Executions | 100 | 100 | 100 | 100 |
| Environment | 500 × 500 pixels | | 500 × 500 pixels | |
| Elimination time | 20000 iterations | | [3000–20000] iterations | |

known as i) dynamicity, ii) time-dependency and iii) uncertainty. Details of the constraints and simulations are explained in the following sections. In these simulations, the environment is a two-dimensional 500×500-pixel space in which, agents, survivors, and the size of obstacles are assumed as 1 pixel. The details of parameter adjustments used in *AEPSO* and the variations of *basic PSO* are demonstrated in Tables 2 and 3.

In the experiments, two environments denoted as Static and Dynamic environments are defined. The details of these environments are as follows:

**Static Environment:** In this environment, survivors and obstacles positions are fixed from the initialization phase. Likewise, survivors' elimination times are set at 20,000 iterations (simple goals) as discussed in Table 2. ($\forall\, s_i \in S,\, v_i^s = 0\ \forall\, s_i \in S, t_i = 20000$).

**Dynamic Environment:** In the dynamic environment, survivors can use stochastic movements. Survivors' velocity is three pixels per iteration ($\forall\, s_i \in S,\, v_i^s = 3$). Survivors use arbitrary directions and renew them iteratively. Survivors have no

**Table 3** *Basic PSO* and *AEPSO*'s variation

| *PSO* type | Details | | |
|---|---|---|---|
| | Inertia weight | Acceleration coefficients | Used equations |
| AEPSO | *(LDIW)*[a]:$w_1 = 0.2, w_2 = 1$ | *(FAC)*[d]:$c_1 = 0.5, c_2 = 2.5$ | $v$:7, $w$:4, $x$:2, $p$:5, $g$:6, $V_{max}$:3 |
| *basic PSO1* | *(LDIW)*:$w_1 = 0.2, w_2 = 1$ | *(FAC)*:$c_1 = 0.5, c_2 = 2.5$ | $v$:1, $w$:4, $x$:2, $p$:5, $g$:6, $V_{max}$:3 |
| *basic PSO2* | *(FIW)*[b]:$w = 0.729844$ | *(FAC)*:$c_1 = 0.5, c_2 = 2.5$ | $v$:1, $x$:2, $p$:5, $g$:6, $V_{max}$:3 |
| *basic PSO3* | *(RANDIW)*[c]:$w \in [0, 1]$ | *(FAC)*:$c_1 = 0.5, c_2 = 2.5$ | $v$:1, $x$:2, $p$:5, $g$:6, $V_{max}$:3 |

[a]Linearly decreasing inertia weight (i.e., Eq. 4)
[b]Fixed Inertia Weight
[c]Random inertia weight
[d]Fixed Acceleration Coefficients

idea of the agents' trajectory and therefore, they can not get close to the agents consciously. Agents should find wandering survivors as fast as possible.

In Table 2, time-dependencies of the tasks in static and dynamic environments are simulated in the third simulation and they are referred to as *TS* and *TD*, respectively.

The experiments investigated in this simulation are presented in Table 4.

### 7.1 First Simulation: Feasibility of Employing *basic PSO* in Static Environment

In this simulation, macroscopic models of *basic PSO* are employed as movement controller and decision makers of simulated agents in survivor-rescuing missions. In this simulation, the effects of entities such as swarm size, parameter adjustments, and population density are investigated in a static environment. Agents are not allowed to leave an area unless they have completed their rescuing task (*ADP1*), and their communication range is set to 5 pixels.

In this simulation, a comparison between the variations of *basic PSO* is presented. The *PSO* variations, in terms of parameter adjustment, are defined as *basic PSO1*, *basic PSO2*, and *basic PSO3*. The details about these variations are discussed in Table 3. The experiments investigated in this simulation are presented in Table 4.

To examine the effects of swarm size, experiments with *basic PSO1* are replicated by 5, 10, 15 and 20 agents for 100 executions. This experiment is referred to as swarm size in the following chapters. Furthermore, a set of experiments are designed to investigate the effects of the survivors' population on the overall performance of three different types of *basic PSO*s (see Table 3). For such an aim, a rescuing mission is designed in which the population of survivors is fifteen, one third of that of the obstacles. Such an experiment is designed to measure the effectiveness of *basic PSO* in environments where the survivors are positioned far from each other. This experiment is referred to as *population density* in the following sections. In previous experiments, due to the high amount of survivors, they are often clustered in a few spots in the environment which gives an advantage to the agents in taking on their rescuing missions. In contrast, in this experiment (population density), the probability of existence of the survivor's cluster is low and this increases the complexity of the missions. The increment of the complexity of the mission is due to the use of fewer number of survivors in the environment which results in their spreading in the wide range of environment.

It is necessary to mention that in our earlier experiments, a version of *basic PSO* without a boundary heuristic, which allowed agents to explore near the boundary lines or even leave the environment, were examined. The results of that version of *basic PSO* showed that agents converged somewhere near the boundary lines or left the environment. Moreover, a version of *basic PSO* with a higher observation-range

**Table 4** Investigated effects on various *basic PSO*s in the first simulation

| Investigated effects | Initial settings | | | | |
|---|---|---|---|---|---|
| | Obstacles | Survivors | Agents | Iteration | Executions/terials |
| Swarm-size | 50 | 50 | 5,10,15,20 | 20000 | 100 |
| Parameter adjustment | 50 | 50 | 5 | 20000 | 100 |
| Population density | 15 | 45 | 5 | 20000 | 100 |

compared to the maximum step-size ($V_{max}$) showed the same behaviour.[20] Therefore, the variation of *basic PSO* used for the comparison is equipped with a simple boundary condition which forces them to relocate aggressive agents somewhere near the environment boundaries. Moreover, observation-radius is considered as three pixels which is less than $V_{max}$ (four pixels per iteration). Fitness function used for these variations of *basic PSO* is based on the achieved performance whenever a survivor is rescued. This simulation and its experiments can be addressed as follows:

- Parameter Adjustment Experiments: $A = \{a_1, ..., a_m\}, m = 5$, and $S = \{s_1, ..., s_k\}$, $k$=*50* and $O = \{o_1, ..., o_n\}$, $n$=*50*, $\forall\ a_i \in A$, $cr = 5$, $\forall\ s_i \in S$, $v_i^s = 0$, $\forall\ s_i \in S$, $t_i = 20000$.
- Swarm Size Experiments: $A = \{a_1, ..., a_m\}$, $m = 5, m = 10, m = 15, m = 20$, $\forall\ a_i \in A, cr = 5, \forall\ s_i \in S, v_i^s = 0, \forall\ s_i \in S, t_i = 20000$.
- Population Density Experiments: $A = \{a_1, ..., a_m\}$, $m = 5$, and $S = \{s_1, ..., s_k\}$, $k$=*15* and $O = \{o_1, ..., o_n\}$, $n$=*45*,$\forall\ a_i \in A$, $cr = 5$, $\forall\ s_i \in S$, $v_i^s = 0$, $\forall\ s_i \in S$, $t_i = 20000$.

### 7.2 Second Simulation: *AEPSO* vs. *Basic PSO*

In this simulation, both static and dynamic environments are simulated in which, survivors, obstacles and agents are located randomly at the initialization phase. The main reasons for implementing such a simulation is to investigate whether *AEPSO* could be employed as a movement controller and decision maker of agents in static environment and to provide a comparison between the variations of *basic PSO* and *AEPSO*. In this environment, a comparison between *AEPSO*, random search, and linear search is presented. Due to the fact that in real world applications the environments are mostly dynamic, *AEPSO*'s performance in such an environment is also examined. In addition, this simulation can be addressed as follows:

- $A = \{a_1, ..., a_m\}$, $m$=*5*, $S = \{s_1, ..., s_k\}$, $k$=*15*, $O = \{o_1, ..., o_n\}$, $n$=*45*, $\forall\ a_i \in A, cr = 5$.
- Static environment: $\forall\ s_i \in S, v_i^s = 0, \forall\ s_i \in S, t_i = 20000$.
- Dynamic environment: $\forall\ s_i \in S, v_i^s = 3, \forall\ s_i \in S, t_i = 20000$.

### 7.3 Third Simulation: *AEPSO* vs. Time Dependency and Random Noise

Since the previous simulation shows *AEPSO*'s advantages and feasibilities as compared to *basic PSO*, the principle idea of implementing this simulation is to investigate *AEPSO*'s performance in both static and dynamic environments affected with time dependency and uncertainty of tasks. Due to the fact that uncertainty and time dependency of tasks are common constraints in real world applications, *AEPSO*'s performance in environments affected with such constraints is examined. In this simulation, time dependency means that each survivor has a specific living time. The living time is randomly initiliased. Uncertainty is assumed as negative or positive random values uniformly distributed between −1 and 1, added to the credit of an area. The task of a simulated agent is to locate the survivors before the living time ends.

[20]Due to poor performances of these two experiments (lower than 40% success rate), these results are omitted from the study.

As in [19, 20], agents are allowed to leave an area without completing the task (*ADP2*) and they are also able to send help request signals. The use of a second area deserting policy (*ADP2*) is due to the fact that *ADP1* which were used in the previous simulation caused a temporary stagnation of an agent and looping of directions and movements. This is due to the fact that *ADP1* forces the agents to continue exploring the areas with positive credits and prevents them from leaving those areas despite the agents' desires to leave the areas. The desire to leave is controlled by the utilities computed from punishments and suspense factors. In contrast, *ADP2* allows agents to leave the areas. This helps to reduce the loop in directions.

In this simulation, agents are able to send help request signal to their neighbors whenever they find themselves in an area with more than one survivors with low living time (near-death survivors). Various communication ranges are used to examine the effectiveness of communication in static and dynamic environments. Suggested communication ranges are 5, 125, 250, and 500 pixels. This simulation can be addressed as follows:

- $A = \{a_1, ..., a_m\}$, $m$=5, $S = \{s_1, ..., s_k\}$, $k$=15, $O = \{o_1, ..., o_n\}$, $n$=45.
- $\forall\, a_i \in A$, $cr \in \{5, 125, 250, 500\}$.
- $\forall\, s_i \in S$,$3000 \leq t_i \leq 20000$.

## 8 Experimental Results

In robotic problems, it is common to use an environment with 500×500 or even 1000×1000 pixels and 200,000 iterations for problem solving tasks. It is also common to use a high number of agent population (approximately between 20 to 100 and even more [31, 34, 37, 45]. Here, five agents with 20,000 iterations are used. In our idea, such constraints are more realistic. In our study, the performance measurement is based on the overall amount of rescued survivors in different experiments. The parameter adjustment and settings in all the simulations have been discussed in the previous section.

In this study, experimental results would be discussed based on several factors. These factors are presented in Table 5.

### 8.1 First Simulation: *Basic PSO*'s Potential in Static Environment

As discussed in Table 4, in this simulation, the experiments are organized based on the effects of parameter adjustment, swarm-size, and population density in a static

**Table 5** Experimental results categorization

| Factors | Remarks | Relevant figures |
|---|---|---|
| The amount of rescued survivors | Figures shows iterations in which survivors are located and rescued | 5a, 6b, 7b, 8a, 9a, 9b |
| The amount of eliminated survivors | Figures shows iterations in which survivors are died | 11, 13b |
| Complete execution (missions) | Figures shows iterations in which various executiones achieved the termination criteria | 5b, 6a, 7a, 8b, 10, 12a, 12b, 13a |

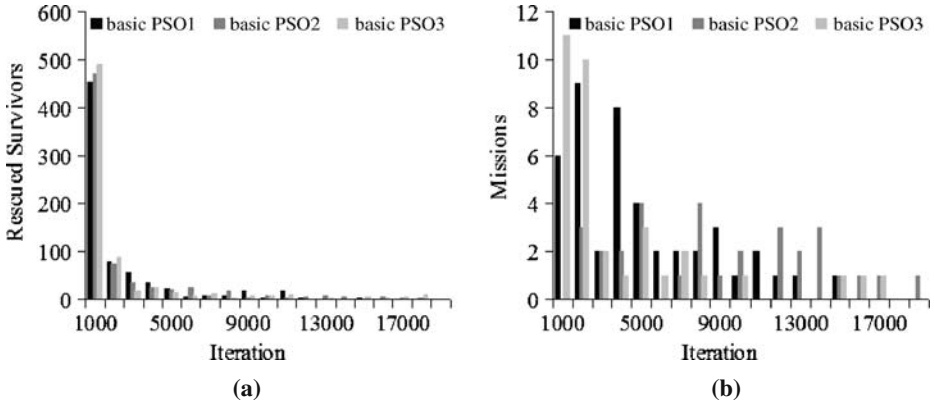



**Fig. 5** The impact of various *basic PSO* in static environment. **a** Rescued survivors. **b** Missions

environment. All experiments presented are based on 100 trials (100 missions with random initial locations).

### *8.1.1 Effects of Parameter Adjustment*

Figure 5a and b show the performance variations of different *basic PSO*s. As discussed in Table 4, in this experiment, *basic PSO1* used Linearly Decreasing Inertia Weight (*LDIW*) and Fixed Acceleration Coefficients (*FAC*), *basic PSO2* used Fixed Inertia Weight (*FIW*) and Fixed Acceleration Coefficients (*FAC*), and *basic PSO3* used Random Inertia Weight (*RANDIW*) and Fixed Acceleration Coefficients (*FAC*). In Fig. 5a, the performance is measured based on the speed of the agent in rescuing the survivors. In contrast, in Fig. 5b, the performance is measured based on the speed of the agents to locate the survivors and finish the mission.[21] The results illustrate that *basic PSO2* performed the worst. This is similar to the results from Vesterstrom and Riget [4]. The low performance in *basic PSO2* is due to its inability to control the search diversity which causes stagnation of particles. In this experiment, *basic PSO1* performed a better search due to the fact that *LDIW* method provides a better control for search diversity as compared to other techniques.

### *8.1.2 Effects of Swarm-Size*

Figure 6a and b shows the effect of swarm size on the *basic PSO*'s performance. In Fig. 6a, the chart shows the achieved results in missions in each period of time (iterations). In Fig. 6b, the chart shows the amount of rescued survivors in each period of time (iterations). Since *basic PSO1* achieved the best performance among others in previous experiment, in this experiment, *basic PSO1* has been used as a movement controller of simulated agents. The swarm size showed a great effect on the overall performance of the team of rescuing agents. It is due to the fact that a team of twenty agents have better chances in terms of communication and

[21] Missions finish either whenever there is no unrescued survivor left or whenever the termination iteration (20,000 iteration in here) is achieved.

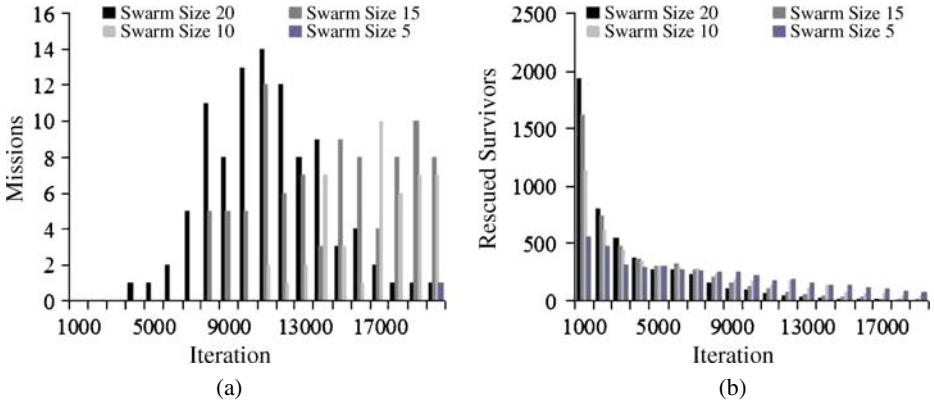



**Fig. 6** The impact of swarm size on *basic PSO* in static environment. **a** Missions. **b** Rescued survivors

cooperation compared to a team of five agents. It is also due to the probability of covering a higher percentage of the environment even with low communication range as in here. As it is mentioned in Tables 3 and 4, to control the value of inertia weight, *basic PSO1* uses *LDIW* method, *basic PSO2* uses *FIW* method, and *basic PSO3* uses *RANDIW*. All of the *PSO*'s variation used in this study uses *FAC* method for setting the acceleration coefficients.

### *8.1.3 Effects of Population Density*

Figure 7a and b show the effects of survivor population on the performance of various *basic PSO*s in static environments. In Fig. 7a, the chart shows the results of the missions in each time period (iterations). In Fig. 7b, the chart shows the amount of rescued survivors in each time period (iterations).

Although the overall performance of all the three methods demonstrate their inability to finish their missions in a predefined time limit, *basic PSO1* performed

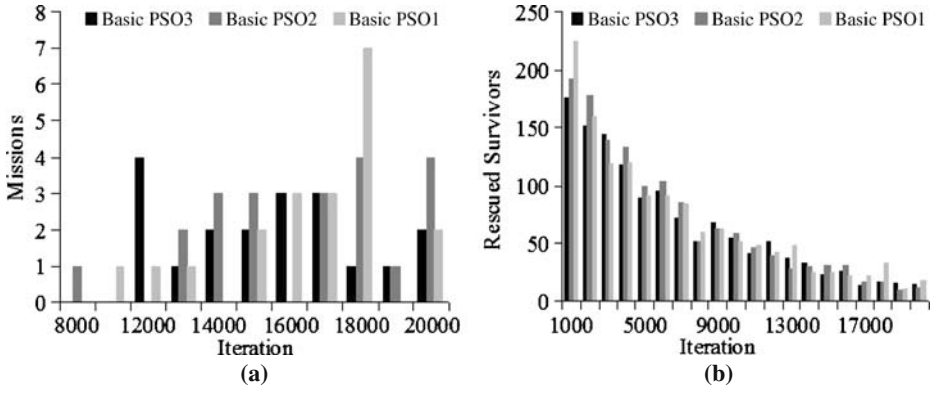



**Fig. 7** The impact of population density on *basic PSO* in static environment. **a** Missions. **b** Rescued survivors

better in terms of the amount of rescued survivors during the missions. The poor variation results of *PSO* are due to their inability to locate the remaining survivors spread in the environment. Such a constraint and disadvantage can be addressed by providing a balance between the exploration and exploitation of the swarm. Although the variation of *PSO* used in these experiments attempt to balance these behaviors, their balancing mechanism frequently ends up in exploring the spots that they should exploit or in exploiting the areas that they should explore. Such a disadvantage persuades us to examine *AEPSO*'s ability in such an environment. It can be concluded that although it is possible to use *basic PSO* in these environments, it is still possible to substitute it with techniques which control the search diversity by providing a better control.

### 8.2 Second Simulation: AEPSO's Potential in Static and Dynamic Environments

Previous experiments revealed that *basic PSO* can be employed as robot controller in suggested environments. Here, we show that it is possible to improve the performance of *basic PSO* by providing better exploration and exploitation. In this experiment, *AEPSO* is compared with *basic PSO*, linear search, and random search. *AEPSO* explores and exploits the search space by utilizing cognitive and social components while random searches and linear searches do not utilize any knowledge. Therefore, *AEPSO* can outperform random search and linear search methods. *AEPSO* is also reliable in dynamic environments since cognitive and social components are dynamically updated.

As in the previous simulation (population density), in this experiment, 5 agents, 15 survivors and 45 obstacles are used. All of the presented experiments are based on 100 trials (executions). The executions end whenever the termination criteria are achieved (mission). A comparison of *AEPSO*'s progression in static and dynamic environments is illustrated in Fig. 8a and b. In Fig. 8a, the progress is presented in terms of the amount of rescued survivors in certain time durations (iterations). The results show a rapid improvement in the early stages followed by a gradual improvement in

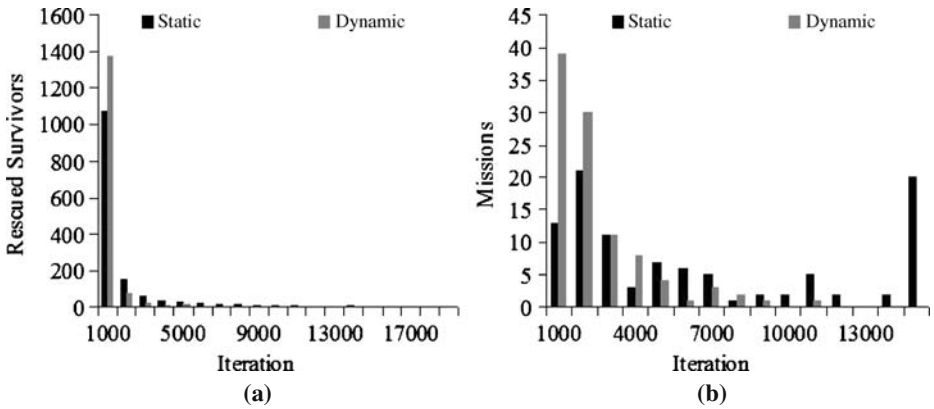



**Fig. 8** The feasibility of *AEPSO* in static and dynamic environment. **a** Rescued survivors. **b** Missions

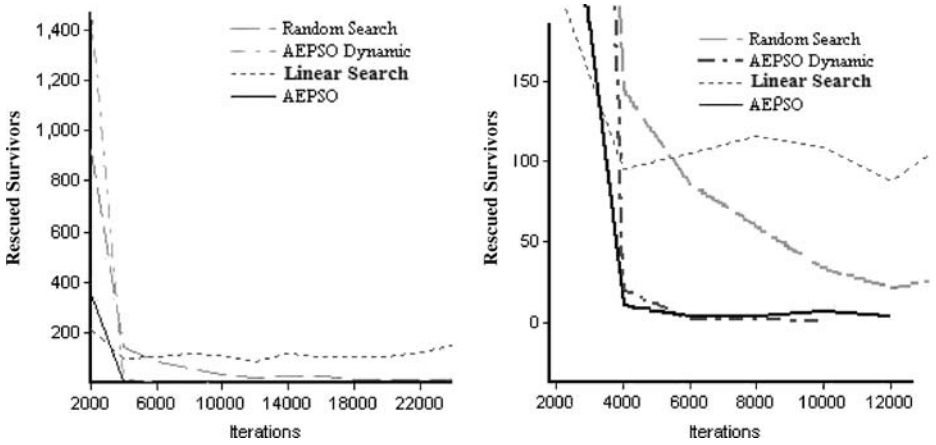



**Fig. 9** Comparison of *AEPSO*'s results in static and dynamic environments, linear search, and random search

the later stages. Figure 8b shows the results achieved in 100 executions in static and dynamic environments with *AEPSO*.

In Fig. 8a and b, the results in dynamic environment show that in the worst execution, the mission was completed by 11,000 iterations (i.e., all of the survivors were rescued). Figures 8 and 9 show the differences of *AEPSO* in static and dynamic environments. Moreover, results from a random search and a linear search method are also shown in this figure.

As the results show, due to dynamic adjustments to social and cognitive components, *AEPSO* rescued more than 80% of the survivors in less than 3,000 iterations (in static environment). Unexpectedly, in a dynamic environment, *AEPSO* rescued more than 95% of the survivors in the first 3000 iterations.

As the results show in Fig. 9a and b, *AEPSO* achieved better results in the dynamic environment in which survivors were moving around. Additionally, in Fig. 9, *AEPSO* shows a high potential for completing the missions in the dynamic environment. The reason is, although dynamic environments have a higher level of complexity as compared to the static ones, survivors' movement helps agents from temporarily-getting-stuck-behind problem. Results show that linear searches and random searches are not reliable in dynamic environment. As the results demonstrate, *AEPSO* outperformed variations of *basic PSO* in static environment because it is able to perform a better local search.

### 8.3 Third Simulation: Time Dependency

As previously discussed, *AEPSO* has the potential to outperform *basic PSO*, linear search, and random search. It is also reliable in dynamic environments in which survivors choose random directions and move iteratively. As time dependency, uncertainty, and communication difficulties are known as common constraints in real world robotic applications, *AEPSO*'s potential in such environments is investigated. The results indicate that *AEPSO* is reliable in time-dependent and uncertain

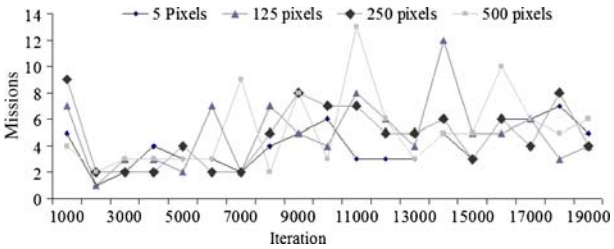



**Fig. 10** Experimental results of *AEPSO* on uncertain static environment with different communication ranges

environments due to its ability to provide better local search due to additional heuristics, *AEPSO*'s ability to give priority to tasks by exploiting areas with lower overall living times first, and also due to the use of help request signal. The results indicate that different communication ranges affect the performance of the *AEPSO* due to its effect on providing sub-swarms and dynamic neighborhood topology.

In this simulation, the results are presented in two groups according to the use of various area deserting policies. As mentioned earlier, in the first group, the first area deserting policy which is denoted by *ADP1* is used. In contrast, in the second group, the second area deserting policy which is denoted by *ADP2* is used. Experiments of both groups are replicated by using various communication ranges (e.g., 500, 250, 125, 5 pixels). The survivors' living times are set to a random value in the range between 3000 to 20000 iterations (time-dependent problem). All of the presented results are based on 100 executions and both of the scenarios are re-examined with a simple noise involved in dynamic and static environments. As discussed in Table 2, in this experiment, 15 survivors, 45 obstacles and 5 agents are used (as in population density experiment where *basic PSO* showed its poorest performance).

Figure 10, describes the impact of different communication ranges on *AEPSO* which is equipped with *ADP2* in an uncertain static environment. As shown, experiences with higher communication ranges showed most of their highest peaks in the first half of the experiment (sometime between 0 to 11000 iterations), while, those who have lower communication ranges have their highest peaks in the second half of the experiment.

Figure 11a and b describe the effects of various communication ranges on the loss of survivors (elimination of survivors) in uncertain environments. The

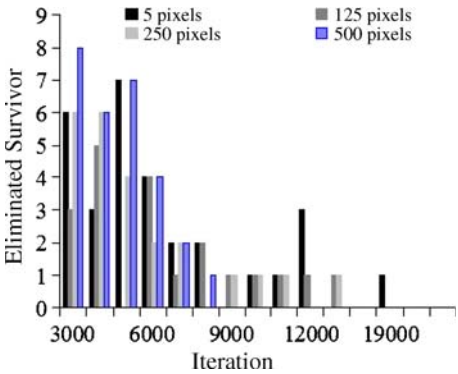



**Fig. 11** Experimental results of *AEPSO* with different communication ranges in uncertain dynamic environment

comparison between the achievements in static and dynamic environments illustrates that *AEPSO* performs better local search in higher communication ranges in which higher cooperation between agents is possible.

The results in both dynamic and static environments show that although in experiments with higher communication ranges agents are unable to rescue survivors with low elimination times, they performed better than others in the rest of the mission and located most of the remaining survivors with high elimination times. It might be due to their lack of knowledge in early iterations. The lack of knowledge in early stages causes an inability to give priority to the tasks accurately. This disadvantage has been addressed with the cooperation between agents in the later stages.

Figure 12a and b, describe the effectiveness of two deserting policies in static and uncertain dynamic environments respectively. The comparison is based on the amount of missions (executions) that were finished before termination iteration. The communication range of an agent is set to 500 pixels and results are achieved from 100 executions with different initializations.

Figure 13a and b show the experimental results of *AEPSO* in various environments (static, static noisy, dynamic, and dynamic noisy) with the communication ranges equal to 250 pixels. It is necessary to notice that in both figures, the second deserting policy (*ADP2*) has been applied. The results are demonstrated based on the impact of various environments and their constraints on the amount of missions that were finished before the termination of iteration and the amount of eliminated survivors.

As the results depict, different communication ranges have major effects on the swarm performance (i.e., low (5 pixels) and high (500 pixels) communication ranges favored in low and high proportion of sub swarms respectively). Although in the communication range of 500 pixels all the agents are able to communicate with each other, the best results are achieved in the range of 250 pixels due to the effectiveness of sub-swarm existence. In addition, *AEPSO* shows reliable results in time dependent (dynamic and static) environments and is able to overcome the noise. (*AEPSO*

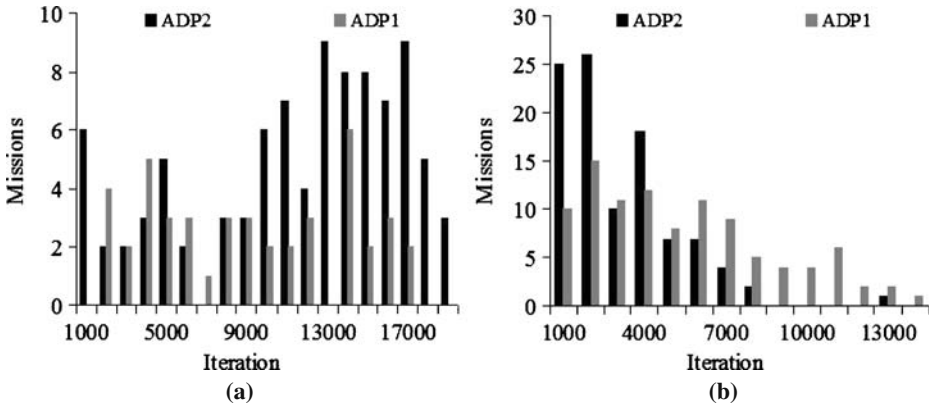



**Fig. 12** Effects of different deserting policies in static and uncertain dynamic environments. **a** Static. **b** Uncertain dynamic

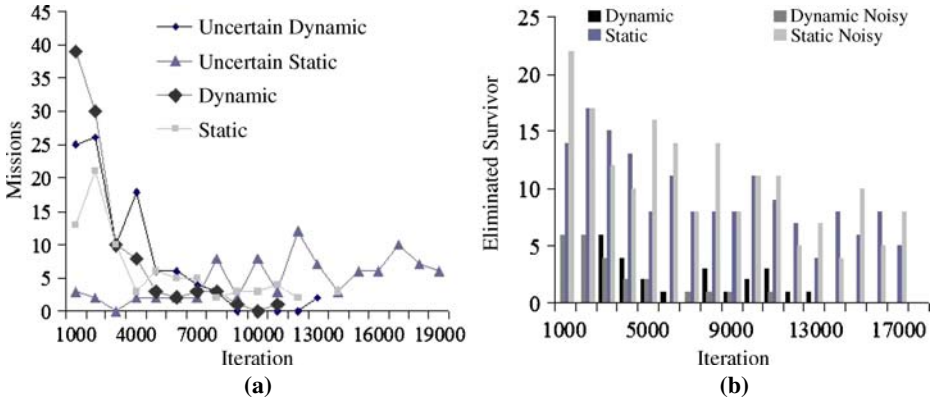



**Fig. 13** Feasibility of *AEPSO* in various environments. **a** Missions. **b** Eliminated survivors

rescued 88% of the survivors in a time dependent, noisy, static environments and 95% of the survivors in a time dependent, noisy, dynamic environments).

## 9 Discussion

In this study, the advantages and disadvantages of various enhanced versions of the *basic PSO* are presented. As mentioned, most of these methods had certain weaknesses. Their poor performances in dynamic, time-dependent, and uncertain environments encouraged us to introduce *AEPSO* as an alternative approach in our study.

### 9.1 The First and the Second Simulations

As other studies and our own results proved, the *PSO* method is not a competitive method for the robotic problem domains due to some of its flaws (e.g., high computing demand and the use of high amount of particles are the drawbacks of *PSO* in robotic domain). In this study, these flaws and weaknesses are addressed with population size problem and the use of maximum iteration. Many previous works hasve suggested the use of swarm-sizes approximately between 20 to 100 agents. Such a high amount for swarm-size is necessary for *PSO* effectiveness in terms of performance. Furthermore, the maximum iteration that is used by other researchers to complete the observation task is 200,000 iterations. In contrast, in this study, we used a 20,000 iteration limitation as our maximum of iterations and our population size is based on using 5 individuals as agents. As the results in static environment display, *AEPSO* achieved a competitive performance as compared to random and linear search methods. Moreover, *AEPSO* outperformed the variations of *basic PSO* which are used for comparison. As the outcomes demonstrate, none of the *basic PSO*s (except in one experiment) used for comparison were able to finish the missions before the termination iteration (20,000 iterations) while *AEPSO* was able to finish 70% and 100% of missions in static and dynamic environments respectively. The closest performance to *AEPSO* is achieved by a version of *basic PSO* which used

20 agents. Furthermore, *AEPSO* show great potential in dynamic environments, in situations where the entire environment changes due to the stochastic movements of the survivors. Based on the results, in a dynamic environment, *AEPSO* located survivors much faster than in a static environment and no loop of direction or temporary stagnation was detected in such an environment. The loop of direction and stagnation problems were caused by the use of suspense factor and first area deserting policy (*ADP1*). The first area deserting policy gives the highest priority to the areas that agents are exploiting inside it while the suspense factor tries to force agents to leave their locating areas. In contrast, in a dynamic environment, due to the iterative movements of the survivors, the agents are able to change the priority of areas iteratively and this helps them to avoid stagnation. The achieved performance is due to *AEPSO*'s capability of providing the balance between exploration and exploitation behaviors. Unlike *basic PSO*, in *AEPSO*, agents only exploit locations which have high probability of achieving performance (i.e., in *AEPSO*, agents only exploit positive-credit-areas). Therefore, *AEPSO* has the advantage in terms of local search.

### 9.2 The Third Simulation

In the third simulation where uncertainty and time-dependency of tasks are examined, *AEPSO* proves to be able to locate survivors within the specified time before their elimination in dynamic and static environments while *PSO*, random search, or linear search are not reliable in such environments. Furthermore, also show that *AEPSO* is also robust to noise (random noise) which helps it to withstand and achieve acceptable performance in uncertain environments. As the results show, the best performances appeared in the communication range of 250 pixels which illustrates the effectiveness of sub swarms on the swarm performance (e.g., in the communication range of 500 pixels, all the members and population of the swarm are involved in the same sub swarm). It is also demonstrated that lower communication ranges which can provide more sub swarms cannot guarantee the achievement of better performances (in the communication range of 5 pixels, each individual particle can be considered as a unique sub swarm). It is due to the existence of individual sub-swarms (in here individual agents) which do not cooperate with each other. Furthermore, experimental results prove *ADP2* efficiency. The low performance of *AEPSO* with the *ADP1* is due to the existence of the loop of directions caused by employed policies. Although such a policy (*ADP1*) forces the agents to remain in positive-credit-areas, it causes them to lose the balance between their behaviors and contributes to their stagnation.

## 10 Conclusion

This study addresses the feasibility of *AEPSO* in robotic scenarios. *AEPSO* has been applied to various robotic search scenarios such as bomb disarming and survivor rescuing. These scenarios address static, dynamic time dependent and uncertain environments. Even though *AEPSO* shows a reliable performance in such environments, it is still not possible to establish it as a general method that can handle the entire robotic search domain. In this study, *AEPSO* showed great potential in

time-dependent and uncertain environments. *AEPSO* was able to outperform the variations of *basic PSO* in static environments. It was due to *AEPSO*'s ability to perform more efficient local searches because it provides a better balance between essential behaviors of the swarm. In addition, *AEPSO* showed a better performance as compared to random and linear search methods. One future direction is to conduct a research to examine the effectiveness of *AEPSO* with physical robots in a real-world application.

## References


1. Kaewkamner, B., Bentley, P.J.: Perceptive particle swarm optimization: an investigation. In: Proceedings of 2005 IEEE Swarm Intelligence Symposium (2005)
2. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC '02), pp. 1671–1676 (2002)
3. Angeline, P.J.: Evolutionary optimization versus particle swarm optimization: philosophy and performance differences. Evolutionary programming VII. In: Lecture Notes in Computer Science, vol. 1447, pp. 601–610. Springer, New York (1998)
4. Vesterstrom, J., Riget, J.: Particle swarms extensions for improved local, multi-modal, and dynamic search in numerical optimization. MSc thesis, Dept. Computer Science, Univ Aarhus, Aarhus C, Denmark (2002)
5. Krink, T., Vesterstrom, J.S., Riget, J.: Particle swarm optimization with spatial particle extension. In: 2002 IEEE World Congress on Computational Intelligence. Congress on Evolutionary Computation (CEC) (2002)
6. Riget, J., Vesterstrom, J.S.: Controlling diversity in particle swarm optimization. In: 2002 IEEE World Congress on Computational Intelligence, Hawaiian, Honolulu. Congress on Evolutionary Computation (CEC) (2002)
7. Riget, J., Vesterstrom, J.S.: Division of labor in particle swarm optimization. In: 2002 IEEE World Congress on Computational Intelligence. Congress on Evolutionary Computation (CEC) (2002)
8. Ratnaweera, A., Halgamuge, S.K., Watson, H.C.: Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans. Evol. Comput. **8**(3), 240–255 (2004)
9. Peram, T., Veeramachaneni, K., Mohan, C.K.: Fitness distance ratio based particle swarm optimization (FDR-PSO). In: Swarm Intelligence Symposium, SIS '03, pp. 174–181 (2003)
10. Parsopoulos, K.E., Vrahatis, M.N.: Particle swarm optimization method in multi objective problems. In: IEEE 2002 ACM Symposium on Applied Computing (2002)
11. Yoshida, Y., Kawata, K., Fukuyama, Y., Takayama, S., Nakanishi, Y.: A particle swarm optimization for reactive power and voltage control considering voltage security assessment. IEEE Trans. Power Syst. (2001)
12. Nomura, Y.: An integrated fuzzy control system for structural vibration. Comput.-Aided Civil Infrastruct. Eng. **22**(4), 306–316 (2007)
13. Ribeiro, P.F., Schlansker, W.K.: A particle swarm optimized fuzzy neural network for voice controlled robot systems. IEEE Trans. Ind. Electron., 1478–1489 (2005)
14. Brits, R., Engelbrecht, A.P., Van Den Bergh, F.: A niching particle swarm optimizer. In: 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002 (SEAL 2002), pp. 692–696 (2002)
15. Brits, R., Engelbrecht, A.P., Van Den Bergh, F.: Scalability of niche PSO. In: Swarm Intelligence Symposium, SIS '03, pp. 228–234 (2003)
16. Zhang, W., Xie, X.: DEPSO: hybrid particle swarm with differential evolution operator. In: IEEE, Systems, Man and Cybernetics (SMCC), Washington DC, pp. 3816–3821 (2003)
17. Chuanwen, J., Jiaotong, S.: A hybrid method of chaotic particle swarm optimization and linear interior for reactive power optimisation. Math. Comput. Simul. (2005)
18. Clerc, M.: The particle swarm—explosion, stability, and convergence in multidimensional complex space. IEEE Trans. Evol. Comput. **6**(1), 57–73 (2002)
19. Atyabi, A., Phon-Amnuaisuk, S.: Particle swarm optimization with area extension (AEPSO). In: IEEE Congress on Evolutionary Computation (CEC), 2007, Singapore, pp. 1970–1976 (2007)

20. Atyabi, A., Phon-Amnuaisuk, S., Ho, C.K.: Effects of Communication Range, Noise and Help Request Signal on Particle Swarm Optimization with Area Extension (AEPSO), pp. 85–88. IEEE Computer Society, Washington DC (2007)
21. Atyabi, A., Phon-Amnuaisuk, S., Ho, C.K.: Effectiveness of a cooperative learning version of AEPSO in homogeneous and heterogeneous multi robot learning scenario. In: IEEE Congress on Evolutionary Computation (CEC 2008), Hong Kong, pp. 3889–3896 (2008)
22. Atyabi, A., Phon-Amnuaisuk, S., Ho, C.K.: Navigating agents in uncertain environments using particle swarm optimisation. MSc thesis, Dept. Information Technology, Univ Multimedia, Cyberjaya, Malaysia (2008)
23. Grosan, C., Abraham, A., Chis, M.: Swarm intelligence in data mining. In: Studies in Computational Intelligence (SCI), vol. 34, pp. 1–20. Springer, New York (2006)
24. Sousa, T., Neves, A., Silva, A.: Swarm optimization as a new tool for data mining. In: Proceeding of the International Parallel and Distributed Processing Symposium (IPDPS'03), Los Alamitos, p. 144.2 (2003)
25. Sousa, T., Neves, A., Silva, A.: Particle swarm based data mining algorithms for classification tasks. IEEE Parallel Comput. **30**, 767–783 (2004)
26. Ahmadabadi, M.N., Asadpour, M., Nakano, E.: Cooperative Q-learning: the knowledge sharing issue. Adv. Robot. **15**(8), 815–832 (2001)
27. Dowling, J., Cahill, V.: Self managed decentralized systems using K-components and collaborative reinforcement learning. In: 1st ACM SIGSOFT Workshop on Self-managed Systems, New Port Beach, pp. 39–43 (2004)
28. Tangamchit, P., Dolan, J.M., Khosla, P.K.: Crucial factors affecting cooperative multirobot learning. In: Conference on Intelligent Robots and Systems, Las Vegas, vol. 2, pp. 2023–2028 (2003)
29. Yamaguchi, T., Tanaka, Y., Yachida, M.: Speed up reinforcment learning between two agents with adaptive mimetism. In: IEEE/RSJ Intelligent Robots and Systems, vol. 2, pp. 594–600 (1997)
30. Yang, C., Simon, D.: A new particle swarm optimization technique. In: 18th International Conference on Systems Engineering (ICSEng 2005), pp. 164–169 (2005)
31. Luke, S., Sullivan, K., Balan, G.C., Panait, L.: Tunably decentralized algorithms for cooperative target. In: Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005), Netherlands, pp. 911–917 (2005)
32. Lee, C., Kim, M., Kazadi, S.: Robot clustering. In: 2005 IEEE International Conference on Systems, Man and Cybernetics (2005)
33. Tan, M.: Multi agent reinforcement learning: independent vs. cooperative agents. In: 10th Int Machin Learning (1993)
34. Chang, K., Hwang, J., Lee, E., Kazadi, S.: The application of swarm engineering technique to robust multi chain robot systems. In: IEEE Conference on Systems, Man, and Cybernetics, vol. 2, pp. 1429–1434 (2005)
35. Bakker, P., Kuniyoshi, Y.: Robot see, robot do: an overview of robot imitation. In: AISB Workshop on Learning in Robots and Animals (1996)
36. Hayas, G., Demiris, J.: A robot controller using learning by imitation. In: 2nd Int. Symp on Intelligent Robotic Systems (1994)
37. Hettiarachchi, S.: Distributed online evolution for swarm robotics. Doctoral thesis, University of Wyoming, USA (2007)
38. Pasupuleti, S., Battiti, R.: The gregarious particle swarm optimizer (G-PSO). In: 8th Annual Conf Genetic and Evolutionary Computation, Seattle, Washington, USA, pp. 67–74 (2006)
39. Bogatyreva, O., Shillerov, A.: Robot swarms in an uncertain world: controllable adaptability. Int. J. Adv. Robot. Syst., **2**(3), 187–196 (2005)
40. Werfel, J., Yaneer, B.Y., Negpal, R.: Building patterned structures with robot swarms. In: Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05), pp. 1495–1502 (2005)
41. Liu, Y., Passino, K.M.: Swarm Intelligence: Literature Overview (2000)
42. Beslon, G., Biennier, F., Hirsbrunner, B.: Multi-robot path-planning based on implicit cooperation in a robotic swarm. In: Proceedings of the Second International Conference on Autonomous Agents, Minneapolis, pp. 39–46 (1998)
43. Qin, Y.Q., Ssun, D.B., Li, N., Cen, Y.G.: Path planning for mobile robot using the Particle Swarm Optimization with mutation operator. In: Proceedmgs of the Third Intemational Conference on Machine Laming and Cybemetics, Shanghai, vol. 4, pp. 2473–2478 (2004)
44. Reggia, J.A., Rodriguez, A.: Extending Self-organizing Particle Systems to Problem Solving. MIT Press Artificial Life (2004)

45. Mondada, F., Pettinaro, G.C., Guignard, A., Kwee, I.W., Floreano, D., Deneubourg, J., Nolfi, S., Gambardella, L.M., Dorigo, M.: Swarm-Bot: a new distributed robotic concept. Auton. Robots **17**(2–3), 193–221 (2004)
46. Martinson, E., Arkin, R.C.: Learning to role-switch in multi-robot systems. In: Autonomous Robots (ICRA'03), vol. 2, pp. 2727–2734 (2004)
47. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, pp. 1942–1948. IEEE, Piscataway (1995)
48. Suganthan, P.N.: Particle swarm optimiser with neighbourhood operator. In: Proceedings of the 1999 Congress on Evolutionary Computation CEC 99, vol. 3, pp. 1958–1962 (1999)
49. Kennedy, J., Spears, W.M.: Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. In: Proceedings of the IEEE Intl Conference on Evolutionary Computation, pp. 78–83 (1998)
50. Fourie, P.C., Groenwold, A.A.: The particle swarm optimization algorithm in size and shape optimization. Struct Multidisc Optim **23**(4), 259–267 (2002)
51. Peer, E.S., Van Den Bergh, F., Engelbrecht, A.P.: Using neighbourhood with the guaranteed convergence PSO. In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium SIS '03, pp. 235–242 (2003)
52. Stacey, A., Jancic, M., Grundy, L.: Particle Swarm Optimization with mutation. In: IEEE Evolutionary Computation, CEC '03, pp. 1425–1430 (2003)
53. Pugh, J., Zhang, Y.: Particle swarm optimization for unsupervised robotic learning. In: Proceedings 2005 IEEE Swarm Intelligence Symposium (SIS), pp. 92–99 (2005)
54. Pugh, J., Martinoli, A.: Multi-robot learning with Particle Swarm Optimization. In: AAMAS'06, Hakodate, pp. 41–448 (2006)
55. Pugh, J., Martinoli, A.: Parallel learning in heterogeneous multi-robot swarms. In: IEEE Congress on Evolutionery Computation CEC'07, Singapore, pp. 3839–3846 (2007)
56. Pugh, J., Martinoli, A.: Inspiring and modeling multi-robot search with Particle Swarm Optimization. In: Proceeding of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007), pp. 332–339 (2007)
57. Zhao, Y., Zheng, J.: Particle Swarm Optimization algorithm in signal detection and blind extraction. In: IEEE Parallel Architectures, Algorithms and Networks, pp. 37–41 (2004)
58. Chang, B.C.H., Ratnaweera, A., Halgamuge, S.K., Watson, H.C.: Particle Swarm Optimization for protein motif discovery. Journal of Genetic Programming and Evolvable Machines, 203–214 (2004)
59. Martinoli, A., Easton, K., Agassounon, W.: Modeling swarm robotic systems: a case study in collaborative distributed manipulation. Int. J. Rob. Res. **23**(4), 415–436 (2007)
60. Agassounon, W., Martinoli, A., Easton, K.: Macroscopic modeling of aggregation experiments using embodied agents in teams of constant and time-varying sizes. In: Autonomous Robots, Hingham, vol. 17, pp. 163–192 (2004)
61. Lerman, K., Galstyan, A., Martinoli, A.: A macroscopic analytical model of collaboration in distributed robotic systems. In: Artificial Life, vol. 7, pp. 375–393 (2001)
62. Kazadi, S., Abdul-Khaliq, A., Goodman, R.: On the convergence of puck clustering systems. Robot. Auton. Syst. **38**(2), 93–117 (2002)
63. Li, L., Martinoli, A., Abu-Mostafa, Y. S.: Learning and Measuring Specialization in Collaborative Swarm Systems, vol. 12(3–4), pp. 199–212. International Society for Adaptive Behavior (2004)
64. Yang, E., Gu, D.: Multiagent reinforcement learning for multi-robot systems: a survey. Technical report, Department of Computer Science, University of Essex CSM-404 (2004)
65. Park, K.H., Kim, Y.J., Kim, J.H.: Modular Q-learning based multi-agent cooperation for robot soccer. Robot. Auton. Syst. **35**, 109–122 (2001)
66. Fujii, T., Arai, Y., Asama, H., Endo, I.: Multilayered reinforcement learning for complicated collision avoidance problems. In: Proceeding of the IEEE International Conference on Robotics and Automation, vol. 3, pp. 2186–2191 (1998)