Turk J Elec Eng & Comp Sci, Vol.18, No.5, 2010, © TÜBİTAK
doi:10.3906/elk-0906-7

Asynchronous particle swarm optimization-based search with a multi-robot system: simulation and implementation on a real robotic system

Salih Burak AKAT[1], Veysel GAZİ[1*] and Lino MARQUES[2†]
[1] *Salih Burak Akat and Veysel Gazi are with TOBB University of Economics and Technology, Department Electrical and Electronics Engineering, Söğütözü Caddesi, No: 43, 06560 Ankara-TURKEY*
e-mail: sbakat@etu.edu.tr,vgazi@etu.edu.tr
[2] *Lino Marques is with University of Coimbra, Department Electrical and Computer Engineering, Institute of Systems and Robotics, 3030-290 Coimbra-PORTUGAL*
e-mail: lino@isr.uc.pt

Abstract

In this article we consider a version of the Particle Swarm Optimization (PSO) algorithm which is appropriate for search tasks of multi-agent systems consisting of small robots with limited sensing capability. The proposed method adopts asynchronous mechanism for information exchange and position (way point) updates of the agents. Moreover, at each (information exchange) step the agents communicate with only a possibly different subset of the other agents leading to a dynamic neighborhood topology. We implement the algorithm using the Player/Stage realistic robot simulator as well as on real KheperaIII robots using experimentally collected realistic data of ethanol gas concentration. Simulation and implementation results show that the algorithm performs well in a sense that the robots are able to move towards and aggregate in areas with high gas concentration around the maximum points of the gas concentration profile representing the environment.

1. Introduction

Searching for one or more targets in an unknown and possibly dangerous (for humans) environment is a task that can be performed by deploying multiple autonomous robots. Equipping the robots (to which we refer as agents in this article) with the necessary sensors and developing efficient navigation and cooperative search algorithms can lead to improving the performance of the system in terms of more effective exploration/coverage

*Their work was supported in part by TÜBİTAK (The Scientific and Technological Research Council of Turkey) under grant No. 106E122 and by the European Commission under the GUARDIANS project (FP6 contract No. 045269).
†His work was supported by the European Commission under the GUARDIANS project (FP6 contract No. 045269).

and decreasing the time of search. In addition to these, there are several other advantages of deploying multi-agent systems (due to their inherent properties) in tasks such as search of an unknown environment compared to the case of using single agent. These advantages include properties such as flexibility and robustness to failures [1].

In order to perform the search task fast and effectively using multi-agent systems, there is a need for an appropriate cooperative search algorithm suitable for multi-agent systems. The algorithm should possibly allow decentralized decision making and parallel operation, dynamic neighborhood topology of communicating agents and asynchronous implementation which are inherent properties of multi-robot systems. In other words, the agents should be able to operate autonomously without a need for a centralized supervisor. Moreover, the overall system should be able to continue its operation even in the case of temporary or permanent communication or agent failures.

The Particle Swarm Optimization (PSO) algorithm [2, 3] is a biologically inspired optimization method which has gained significant popularity in the past decade. It is a non-gradient, direct search based optimization strategy in which a set (a population) of N possible solutions (referred to as a swarm of particles) is iterated in parallel in search for the best solution in a multi-dimensional (n-dimensional) space (or region/domain). At each iteration each particle updates its velocity based on its past velocity (momentum component), its past best position (cognitive component), and the past best position of its neighborhood (social component). Due to its inherent decentralized nature and the fact that it does not require gradient information and relies only on the function measurements at the current positions of the particles, it seems suitable for multi-agent search applications.

There have been works on investigating search strategies inspired from Particle Swarm Optimization for multi-agent systems. Doctor and his colleagues studied a multi-robot search application involving one or more targets [4]. Their study is focused on finding optimal parameters for the PSO algorithm, including the inertia weight parameter (w) and upper bounds of learning coefficients ($\bar{\varphi_1}$ and $\bar{\varphi_2}$), in order to perform a target search task efficiently. They use a 2-level hierarchy in which the PSO in the inner level solves the problem of finding the locations of target/targets, whereas the PSO in the outer level determines the optimal set of parameters for the inner level PSO. Hereford [5, 6] considered a distributed PSO for robot search application which is somewhat similar to the algorithm we consider here. His emphasis was on simplicity and decreasing the communication burden in the system and to achieve scalability of the algorithm to large number of robots. Pugh and Martinoli [7, 8] worked on adapting Particle Swarm Optimization algorithm to multi-agent search applications. In [7] they considered two techniques as adapting multi-agent search application to PSO and adapting PSO to multi-agent search application in order to create a microscopic model. They considered the cases in which: (i) agents know their global positions and; (ii) agents rely on their local knowledge. The work in [7] contains important ideas, some of which have similarities with the work in this paper, and nice comparison between similarities and differences of PSO and multi-robot search. However, they used a synchronous version of the PSO algorithm. Moreover, no implementation on real robots was done in [7]. In [8] they used a bacteria foraging inspired search algorithm and for better performance they tuned its parameters using the PSO algorithm. Marques and his colleagues [9] presented a PSO inspired search method in order to detect odor sources across large search spaces. They formulated the odor finding problem and developed a model for instantaneous odor concentration at ground level. Moreover, they compared the PSO inspired search method with other gradient related methods and other evolutionary strategies and observed that, in their setting which has unstable environment with high

turbulence, the PSO inspired search method is more successful compared to the other considered strategies. There is also related recent study [10] applying PSO inspired multi-robot search for finding targets wireless signals (possibly from a cell phone). They use the RSS signal strength as the fitness function to be maximized in order to locate the targets. However, no implementation on real robots is done in [10]. Moreover, notice that the algorithm in this paper was developed and the results were obtained earlier than those in [10].

In [11, 12] a framework for PSO with dynamic neighborhood topology is presented and implemented on several benchmark functions. The results obtained show that PSO with dynamic neighborhood topology is a viable variation of the PSO algorithm which may have advantages in applications such as the multi-robot search problem considered in this article. Another variation of the PSO algorithm which might have important advantages in distributed systems operating in parallel (such as multi-robot systems) is one with totally asynchronous updates. A related study on such a variation was performed in [13, 14]. Such an implementation removes the need for global iteration counter and synchronization between the particles (robots) and may improve the performance of the system (since the particles/agents do not have to wait for the other agents) and prevent the system from getting stalled because of possible communication or agent failures.

Note that there are also other studies which employ dynamic neighborhood [15] or parallel [16] or parallel and asynchronous [17] operation. The dynamic neighborhood in [15] is based on organizing the swarm in subswarms and after a predefined number of iterations redefining the subswarms and the neighborhood topology employed in this article is completely different. Similarly, the implementation in [16] is parallel but synchronous and the asynchronism employed in [17] is of centralized nature. In particular, in [17] there is a master computer performing the PSO updates, whereas the slave computers perform only the function evaluations in possibly asynchronous manner. The implementation in this article is based on completely different philosophy and is totally asynchronous and decentralized. For more discussion on these and reference to other relevant work the reader may consult [11, 12, 13, 14].

Inspired by the works in [11, 12, 13, 14], in this article we consider a system consisting of multiple robots deployed in a search task using Particle Swarm Optimization based decision making process and position updates. The robots are allowed to operate asynchronously and to exchange information using dynamic neighborhood topology. To the best of our knowledge PSO inspired robotic search in the manner we consider here (dynamic neighborhood and asynchronous operation) and its implementation on real robotic system has not been considered so far in the literature. Since these properties are inherent in multi-agent systems we believe that the proposed method, allowing asynchronous operation and dynamically changing communication topology, is more suitable for multi-robot implementations.

2. Problem Formulation

In this article we consider a system consisting of N mobile robots (agents) moving in $\mathbb{R}^2$ with continuous-time non-holonomic dynamics given by

$$\begin{aligned} \dot{x}_i(t) &= \bar{v}_i(t)\cos(\theta_i(t)), \\ \dot{y}_i(t) &= \bar{v}_i(t)\sin(\theta_i(t)), \\ \dot{\theta}_i(t) &= w_i(t), \end{aligned} \tag{1}$$

where $x_i(t)$ and $y_i(t)$ are the Cartesian coordinates, and $\theta_i(t)$ is the steering angle of the i^{th} agent at time t. The control inputs of the i^{th} agent are the linear speed (not velocity) $\bar{v}_i(t)$ and the angular speed $w_i(t)$. The robots are required to perform a search in an unknown environment. Each point in the environment is assumed to have a particular potential value which, in this article represents experimentally collected ethanol gas concentration. However, in general, depending on the problem under consideration, it can represent other entities such as different type of odor, chemical plume or smoke concentration, temperature or light intensity etc. We call this potential the resource profile and use PSO based optimization strategy to plan the motions of the robots with the objective of locating the extremum (minimum or maximum) points of the resource profile. Again, depending on the application, these points can represent the source of odor, smoke/fire, heat or light. We assume that the robot is equipped with the necessary sensors in order to be able to measure the value of the environmental resource profile.

From the perspective of PSO inspired search robot i constitutes particle i. Given the robot is at its k^{th} way point $p_i(t_k) = [x_i(t_k), y_i(t_k)] \in \mathbb{R}^2$ at time t_k (iteration k for the robot/particle), its (desired) next way point $p_i(t_{k+1})$ is calculated using the PSO algorithm. In other words, PSO is used for higher level path planning for determining the way points that the robot should visit. In order to move the robot from the k^{th} way point $p_i(t_k)$ to the $(k+1)^{\text{th}}$ way point $p_i(t_{k+1})$ we use artificial potential functions for low level control. In particular, we use the straight path between the two points and require the robot to move along the vector $\bar{p}(t_k) = p_i(t_k) - p_i(t_{k+1})$ to reach $p_i(t_{k+1})$ (with collision avoidance). With this objective, we use a quadratic attractive potential function and require the robot to move along its negative gradient. Similarly, in order to avoid collisions between robots we use a repulsive potential function which is activated when the distance between two robots becomes less than a predefined constant value d. Let us denote the gradient of the attractive potential with $G_{ai}(t)$ and the gradient of the repulsive potential for robot i due to robot j (in its vicinity) as $G_{rij}(t)$ at time $t \in [t_k, t_{k+1})$. In this article, we determine their values using the relations

$$G_{ai}(t) = -a\bar{p}_i(t),$$

and

$$G_{rij}(t) = \begin{cases} b\bar{p}_{ij}(t)\left(\frac{1}{\|\bar{p}_{ij}(t)\|^2} - \frac{d}{\|\bar{p}_{ij}(t)\|^3}\right), & \|\bar{p}_{ij}(t)\| \le d, \\ 0, & \|\bar{p}_{ij}(t)\| > d, \end{cases}$$

where $\bar{p}_i(t) = \big(p_i(t) - p_i(t_{k+1})\big)$ and $\bar{p}_{ij}(t) = \big(p_i(t) - p_j(t)\big)$. Here, $p_i(t) = [x_i(t), y_i(t)] \in \mathbb{R}^2$ represents the current position of the robot at time $t \in [t_k, t_{k+1})$, $p_j(t) = [x_j(t), y_j(t)] \in \mathbb{R}^2$ represents the current position of robot j, which is in the vicinity of robot i, at time $t \in [t_k, t_{k+1})$ and $p_i(t_{k+1})$ represents the next way-point of the robot in the search space. The constant parameter $a > 0$ is an attraction coefficient, whereas the constant parameter $b > 0$ is a repulsion coefficient. Note that with this configuration there is a linear attraction force for each robot towards its respective way-point in the search space. In contrast, the nonlinear (and unbounded) repulsion force is activated only if there are robots in the vicinity of robot i at a distance smaller than a predefined distance d and is used to avoid robot to robot collisions.

Using the above, the total potential (field) $G_i(t)$ for robot i is determined as

$$G_i(t) = G_{ai}(t) + \sum_{j=1}^{N} G_{rij}(t), \tag{2}$$

where although the summation of the repulsion terms is over all the other robots, only those which are within the detection range of robot i actually effect the value of $G_i(t)$. Moreover, we use the local coordinate system of the robot for calculating the attraction and repulsion forces.

Since the robots are non-holonomic (i.e., they cannot move in the direction of the axis connecting the two main wheels) and since their orientations initially may not necessarily be along the vector potential, we define the desired direction of motion as

$$\theta_{id}(t) = \text{atan2}\Big(G_{yi}(t), G_{xi}(t)\Big), t \in [t_k, t_{k+1}),$$

where $G_{xi}(t)$ and $G_{yi}(t)$ represent the x and y components of the potential field in equation (2), respectively. Then, for the orientation dynamics of the robot we use the simple proportional controller

$$w_i(t) = -\alpha\Big(\theta_i(t) - \theta_{id}(t)\Big), t \in [t_k, t_{k+1}), \tag{3}$$

where $\theta_i(t)$ represents the current orientation of the robot at time t and $\alpha > 0$ is the proportional gain. For the linear speed controller we use

$$\bar{v}_i(t) = \min\{\|G_i(t)\|, v_{\max}\}, t \in [t_k, t_{k+1}), \tag{4}$$

where $v_{\max}$ is an upper bound on the linear speed of the robot. Note that it might be possible to design more sophisticated and effective controllers as well. However, in this article the emphasis is on the PSO based search algorithm, not on the low-level controller development.

3. Asynchronous PSO

There are various possible implementations of the Particle Swarm Optimization algorithm. Some of them suffer from the so-called explosion problem and need to employ a bound on the speed of the particles. In this article we use the PSO version that uses a "constriction coefficient" proposed by Clerk and Kennedy in [18] in which at the k^{th} iteration, which corresponds to time t_k for a given robot, the update for the way points for particle (robot) i is given by

$$\begin{aligned} v_i(t_{k+1}) &= \chi\Big[v_i(t_k) + \varphi_1^i(t_k)\Big(b_i(t_k) - p_i(t_k)\Big) + \varphi_2^i(t_k)\Big(g_i(t_k) - p_i(t_k)\Big)\Big], \\ p_i(t_{k+1}) &= p_i(t_k) + v_i(t_{k+1}), \end{aligned} \tag{5}$$

Here, $p_i(t_k) \in \mathbb{R}^2$ represents the position (way point) of the i^{th} particle at time t_k (the estimation of this particle about the minimum/maximum point of the function being optimized at time t_k corresponding to iteration k of the PSO algorithms), $b_i(t_k) \in \mathbb{R}^2$ represents the best position *pbest* of the i^{th} particle until time t_k, $g_i(t_k) \in \mathbb{R}^2$ represents the best position *gbest* of the neighborhood of the i^{th} particle until time t_k. The value $p_i(t_{k+1}) \in \mathbb{R}^2$ which is calculated at the end of the iteration is the next (desired) way point to which the robot should move. The learning coefficients $\varphi_1^i(t_k) \in [0, \bar{\varphi}_1]^2$ and $\varphi_2^i(t_k) \in [0, \bar{\varphi}_2]^2$ are two dimensional uniform random vectors. At each iteration these random vectors respectively determine the relative significance/weight

of the cognitive and social components in the iteration. The constant parameter $\chi > 0$ is the constriction parameter that prevents the explosion behavior, i.e., it prevents particles having high velocity values leading to their scattering in the search space. In the implementations which correspond to (5) for efficient performance and prevention of the explosion behavior, the components of the $\varphi_1^i(t_k)$ and $\varphi_2^i(t_k)$ learning coefficient vectors must satisfy the relation [18]

$$\varphi_{1j}^i(t_k), \varphi_{2j}^i(t_k) \in [0, 2.05], j = 1, \ldots, n; i = 1, \ldots, N, \tag{6}$$

where N is the number of particles (i.e., robots in this case) and n represents the number of search dimension (i.e., $n = 2$ in the case here). The constriction parameter $\chi > 0$ can be calculated using the relation (refer to [18])

$$\chi = \begin{cases} \frac{2\kappa}{\varphi - 2 + \sqrt{\varphi^2 - 4\varphi}}, & \text{if } \varphi > 4, \\ \kappa, & \text{otherwise.} \end{cases} \tag{7}$$

In this equation $\varphi = \bar{\varphi}_1 + \bar{\varphi}_2$ and $\kappa \in [0, 1]$. In this article, we calculated the constriction factor $\chi > 0$ based on the $\bar{\varphi}_1$ and $\bar{\varphi}_2$ values above ($\bar{\varphi}_1 = \bar{\varphi}_2 = 2.05$ so $\varphi = 4.1$) and choosing the coefficient $\kappa = 1$.

There are other possible alternative implementations of the PSO algorithm involving different parameters, such as different coefficients and bound particle velocities. However, since here the emphasis is on robot search, and not on comparing these alternative implementations, we will stick to the above version in this work.

In its basic form, iteration in (5) is performed synchronously after all the particles have performed their function evaluations and exchanged information. Although, even in its basic form, the method seems to be appropriate enough for robot search applications, we propose some modifications of it to improve its performance and to overcome some shortcomings that might arise. These modifications are in the line of the works in [11, 12, 13, 14] and include decentralized and asynchronous implementation (much different from the ones considered in the literature) and dynamic information exchange topology between the robots (i.e., particles) in the system.

First of all, note that multi-robot systems are naturally distributed and decentralized decision making and operation are among their desired properties for higher levels of robustness and flexibility. Moreover, due to the decentralized/distributed nature, multi-robot systems operate in an asynchronous manner. Furthermore, the sensing and communication capabilities of the robots are usually limited, which results in time dependent interactions (since only those robots which are within each others sensing or communication range can interact). Therefore, direct implementation (without modification) of the PSO algorithm for search in multi-robot systems may result in unsatisfactory performance. This is because, first of all, the robots cannot instantaneously jump to their next way points and it may take different amounts of time for the robots to reach their respective way points. For this reason, in classical PSO implementations the robots which arrive earlier than the others have to wait for all the robots to arrive to their respective way points before exchanging information with the objective to update the global best fitness value of the neighborhood (*gbest*) and to move to the next iteration of the PSO algorithm. Moreover, for large search areas the respective way points of the robots may be far from one another and the distance between robots may exceed the communication range. This may lead some robots to become unable to communicate with one another and therefore in classical PSO implementations the system may stall since, in order to move to the next iteration, the robots need to obtain information from all their neighbors. The situation may become even worse in the presence of temporary or permanent communication or

agent failures. To overcome these shortcomings, motivated by the works on asynchronous PSO and PSO with dynamic neighborhood in [11, 12, 13, 14], we modify the PSO algorithm as described in the pseudocode given in Algorithm 1.

1. Pseudocode describing the algorithm presently discussed.

```
Initialize pbest and gbest (and all other variables).
Calculate the first way point randomly.
while (Stopping criteria is not satisfied) do
  while (Agent has not arrived to its way point) do
    Move towards the desired way point.
    Read fitness (gas concentration) data from the current position.
    Update pbest.
    if (Information received from other agents) then
      Update pbest_other.
    end if
  end while
  Broadcast own pbest.
  if (pbest_other > gbest or pbest > gbest)) then
    Update gbest.
  else
    Use previous gbest.
  end if
  Calculate a new way point using Equation (5).
end while
```

As one can see from the algorithm in Algorithm 1, while a robot is moving towards its respective way-point in the search area, it continuously listens for information from other robots (since, if other robots arrive at their respective way-points earlier, they broadcast their best fitness values achieved until that time) and later uses this information for updating the global best fitness value. After arriving at the desired way-point, the robot broadcasts its best fitness value achieved until arriving at the desired way-point. Then using the information gathered by itself and received from the other robots/particles, it performs an update of its next desired position (way-point) based on its current velocity, its best position, the global best position (which is extracted from the information obtained from the other robots) - see equation (5). If there are no other robots which have arrived at their respective way-points earlier than the robot under consideration (meaning that it has not received any information from the other robots yet), the robot continues its update based on its own measurements and the old information which it had obtained from the other robots in the past. In other words, it does update the global best based only the information that it has sensed during its motion and continues its operation. Since the robots may arrive at their respective way-points at different time instants, the way-point updates are performed asynchronously. Moreover, the set of robots from which a given robot receives information at each iteration of the PSO algorithm may change leading to a time-varying (dynamic) neighborhood topology. In the basic form of the PSO algorithm the neighborhoods of the particles are static/fixed, i.e., particle neighborhoods remain stationary throughout all iterations of the algorithm.

4. Simulation and implementation results

The Asynchronous Particle Swarm Optimization-based search method discussed in the preceding sections is first tested using the Player/Stage realistic robot simulator. Player provides an interface to the robot's sensors and actuators over an IP network. A client program sends commands to Player over TCP/UDP sockets and reads data from sensors. Stage provides the simulation environment in order to test the developed algorithm. The simulation environment provides movement of mobile robots in a two dimensional environment and various sensor models [19].

Simulations and implementations are performed in an obstacle free environment using experimentally collected realistic data. The data was gathered inside an enclosed environment with 3-by-4 meters area and 0.5 meter height. This environment was weakly ventilated through an opening in one corner and a system composed by four 12 centimeters diameter fans able to generate a flow ranging from 0 to 1500 lpm (liter per minute) each in the other corner (see Figure 1). There were three gas sources in the set-up placed on the ceiling of the environment in the following locations: G1 (2.25, 0.625), G2 (0.5, 2), and G3 (2.25, 3.45). These gas sources were obtained passing a controlled airflow through ethanol bubblers. The flow was controlled by individual SMC proportional valves. The ethanol concentration inside the environment was acquired with a sensor network composed by twelve Figaro TGS2600 gas sensors. These sensors can detect ethanol vapor starting from few parts per million. Each gas sensor was mounted on a small PCB board with the necessary signal conditioning circuit. The output from these sensors was acquired by two Microchip PIC18F4431 microcontrollers interfacing to a PC through an RS485 shared bus, as represented in Figure 1. The continuous distribution of concentrations was estimated with a kriging estimator.

Given the experimentally collected data, the objective is to determine the areas of high gas concentration. Figure 2 shows the plot of the estimated continuous distribution of the experimentally collected concentration (the environmental gas profile) used in this study. This profile is the instantaneous odor concentration at a given time instant in the 3-by-4 bounded area mentioned above.

For the simulations in Player/Stage, 6 Pioneer 2-dx robots were used.[1] The control inputs, the linear speed ($\bar{v}_i(t)$) and the angular speed ($w_i(t)$), are applied using the controllers discussed in the preceding sections. Robots are equipped with 16 ultrasonic range detectors located at their front, left, right, and rear sites. Robots are not equipped with the devices that provide global position information to them and rely on their position odometry information only throughout the search. However, note that the objective here is not to test the effect of global or local positioning of the robots and the odometry errors on the performance of the algorithm. The proposed algorithm relies on the assumption that a proper positioning is available. It is also assumed that the robots are equipped with necessary sensors to get the potential (i.e., ethanol concentration) value information of each grid in the search area.

Initially the robots are located near the area entrance, designated point (0,0) in the cartesian plane. The first way-points in the search area are generated randomly for the robots and each robot starts to move towards its initial way-point. This helps the robots to localize themselves relative to their start positions without a need for global/absolute positioning information. (Some errors will be present due to the fact that all of the robots cannot start their search at the entrance point rather close to this point and this will cause positioning errors at the beginning in terms of global positions.) Moreover, it models the realistic situation in which the area to

[1] In the Player/Stage realistic simulator the Pioneer 2-dx robots were already modeled (i.e., drivers already existed for the robots).

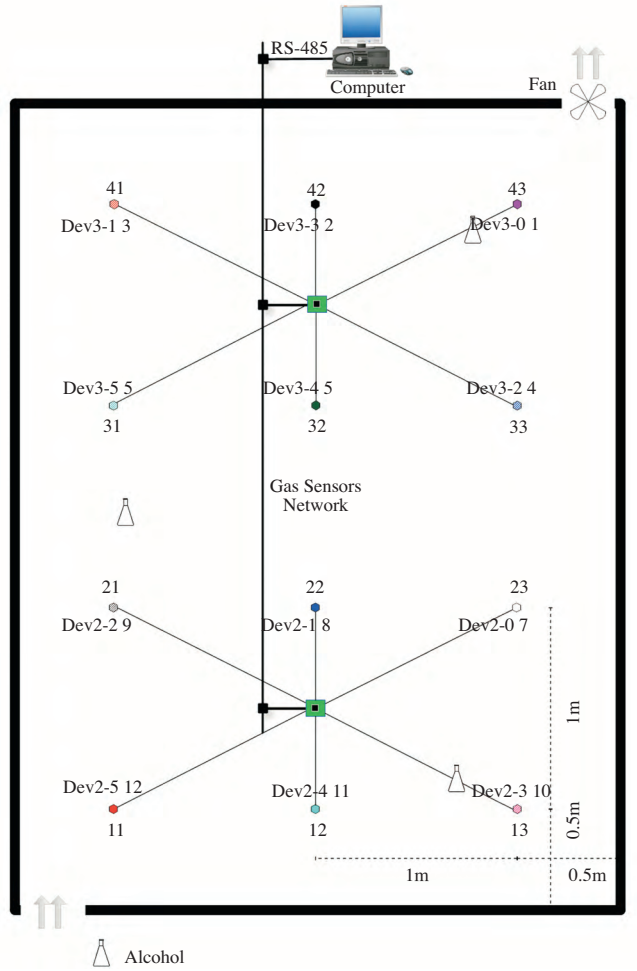

Figure 1. Data collection environment setup.

be searched has only a single entrance (such as a building to be searched). The ultrasonic sensors, which the robots are equipped with, are used to measure inter-robot distances and to prevent robot to robot collisions. In particular, their readings are used for calculating the repulsive potential. For the PSO algorithm the upper bounds for the social and cognitive learning coefficients ($\bar{\varphi}_1$ and $\bar{\varphi}_2$) is chosen as 2.05 and the constriction factor is specified as 0.7298 by exploiting the relation in equation (7).

Figure 3 show the way-points that the robots visited and the trajectories of the robots at the search space respectively for an example run. The "X" symbol represents the random first way-points of the robots (which are effectively the initial positions of the particles of the PSO algorithm), the "O" symbol represents the final positions of the robots, and the curves represent the trajectories of the robots.

The stopping criteria for the robots is determined based on their velocity vector in equation (5). For that purpose we define a small threshold v_{stop} which corresponds to a distance smaller than the size of the robot. Then, if the calculated velocity vector $v_i(t_{k+1})$ in (5) satisfies $\|v_i(t_{k+1})\| < v_{\text{stop}}$ for several iterations

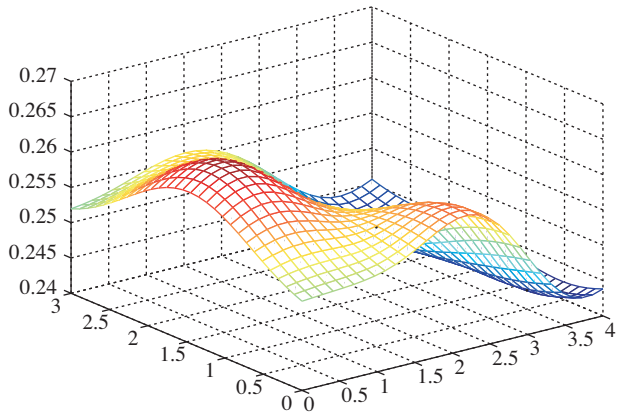

Figure 2. Resource profile.

implying that the calculated new way-points for the robot will be very close to its current way-point the robot decides to stop. This is intuitively reasonable since the robot is almost not moving and has already effectively stopped. Such situation arises when the current position $p_i(t_k)$ is almost equal to the personal best $b_i(t_k)$ and the neighborhood/global best $g_i(t_k)$ and, intuitively, is a good place to stop. The way-points of the robots are shown as "dots" on the contour maps of the profile (see Figure 2 for the profile, while the contour maps are shown in Figure 3). Due to robots physical dimensions, position odometry errors due to initial locations of the robots in the search area, and repulsion forces between them (which are used to prevent robot to robot collisions), it is a fact that all the robots cannot simultaneously reach the same point (i.e., the global maximum point). Rather they congregate around it. Since each robot updates its position and velocity vectors asynchronously and there is no global iteration counter that synchronizes these updates, each robot finds the target in different number of iterations. Figure 4 shows the average distance of robots to the target for each iteration. The vertical lines represent the standard deviation of the distance of robots to the target. The average distance to the maximum and the standard deviation decreases as time progresses and the robots perform their PSO updates. Since the stopping criteria for different robots can be satisfied at different time instants the robots finish their search task at different iterations, and those robots which have finished remain in the same position for the rest of the simulation. It took 30 iterations for the robot that finished last. Therefore, the iteration axis in the figure is up to 30 iterations. From the plot one can observe that the average number of iterations is about 20 iterations. Note also that in both the simulations and in the implementations discussed below it is assumed that the agents know the size of the search area. Moreover, we apply a projection algorithm in order to prevent the robots from going out of the boundary of the search region. In other words, if at a given iteration the PSO algorithm generates a way point which is outside the search region, the robot projects it within the search region and replaces it with the projected point before starting its motion towards the way point.

Implementation of the proposed strategy with real robot platform are performed in an experimental arena with dimensions 3.40 m $\times$ 2.40 m $\times$ 15 cm and no obstacles (shown in Figure 5). Since robots do not have sensors to detect/measure ethanol gas concentration in the search area, the search area is divided into equal square grid cells and a virtual gas concentration value is assigned to each grid according to the instance of experimentally collected realistic data of ethanol gas concentration in Figure 2. For this implementation three

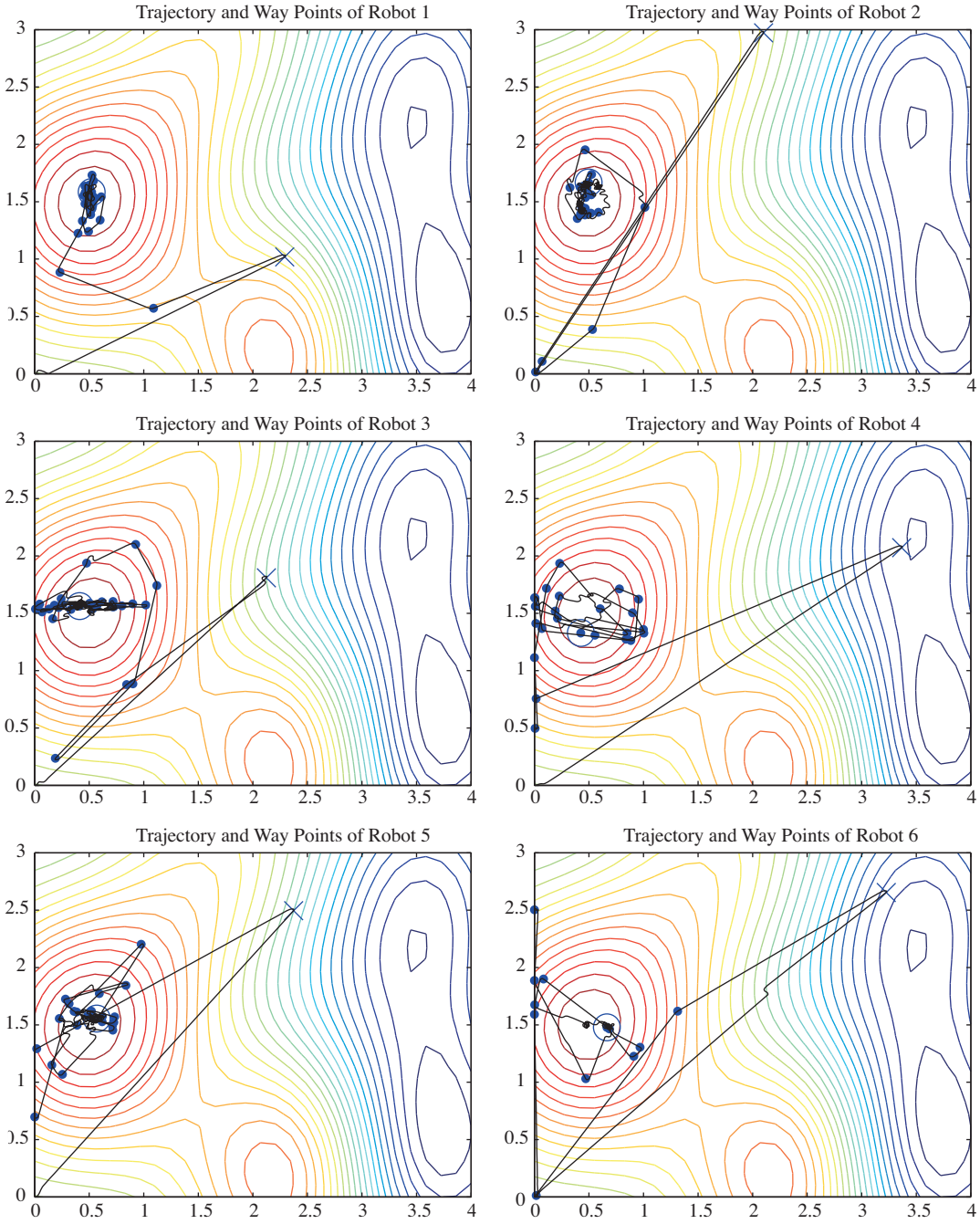

Figure 3. Trajectories and way points of the robots.

KheperaIII robots are used. Robots have Intel PXA255 processors working at 400 MHz. Movement of the robots are provided by 2 brushless DC servo motors. The robots are equipped with 9 infrared and 5 ultrasound distance measurement sensors in their periphery. These sensor can be used for collision avoidance. Each motor

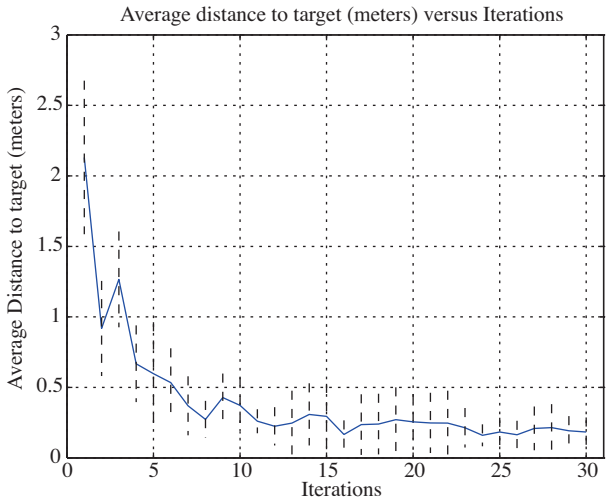

Figure 4. Average distance to the maximum.

is driven by its own PID controller implemented on a PIC18F4431 microcontroller and these microcontrollers are also used for collecting the information from the optical encoders connected to the motors and calculating the odometry information. The motor control blocks act as slave devices on the I2C bus while communicating with a master DSPIC30F5011 microcontroller. In order to determine/measure the sensor readings the DSPIC30F5011 microcontroller working at 60 MHz is used and the sensors communicate with the microcontroller via the I2C bus.

Figure 5. Application area.

Figure 6 shows snapshots of the instant positions of the robots at the arena at 1st (initial locations), 717th, 1283th, and 2265th (final positions) frames of the recorded video of an example run.

As in the simulations, the robots are initially placed close to entrance of the search area, close to the $(0, 0)$ point (see Figure 6). Odometry information of the robots are used for localization during search. The

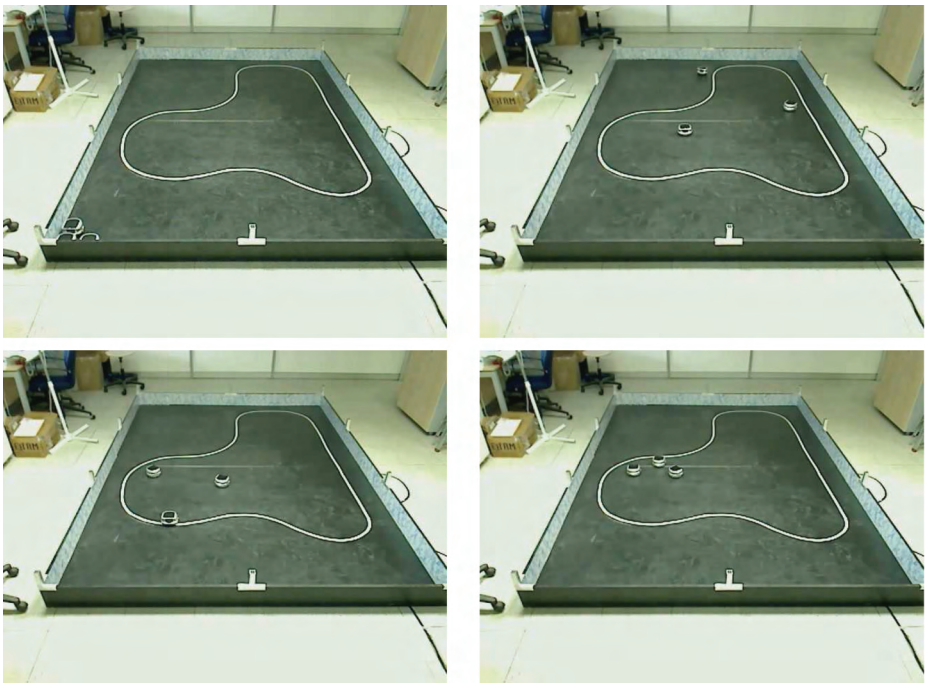

Figure 6. Snapshots from the implementation with KheperaIII robots at 1st, 717th, 1283th, and 2265th frames respectively.

measurements/readings of the infrared sensors are used for calculation of repulsive potential function and collision avoidance. Figure 7 shows the way-points visited and the trajectories of the robots on the contour map of the resource profile (i.e., the ethanol gas concentration map) and average distance to maximum, respectively.

Similar to simulation results with the Player/Stage simulator, in the implementation on real robots as well the robots manage to move towards and aggregate in regions with high gas concentration. In particular, they aggregate in the vicinity of the global maximum.

We would like to also mention here that we have not compared the errors between positioning based on odometry (which we use here) and a global/absolute positioning system. One would expect that large localization errors will lead to unsatisfactory performance. It is worth emphasizing once more that the objective here is not to study the effect of localization on the performance of the algorithm, but to show that it is a viable search algorithm which shows acceptable performance. Note also that although we have presented the results of only one case we have performed other simulations and implementations as well. To further improve the implementation performance of the algorithm, in addition to having accurate localization, improving the navigation/control and collision avoidance algorithm of the robots can be considered (although not in the scope of this article). This is because the potential function methods usually suffer from the so-called local minima problem and such problems might effect the overall performance of the algorithm as well.

One may think that a simple gradient-decent algorithm may work for this application. However, note that gradient information is not available. Only the present value of the concentration is available to the robots as they perform search based on sampled data. In other words, even though the overall map of the gas concentration is available for plotting to the observer, it is not available to the robots during search as they

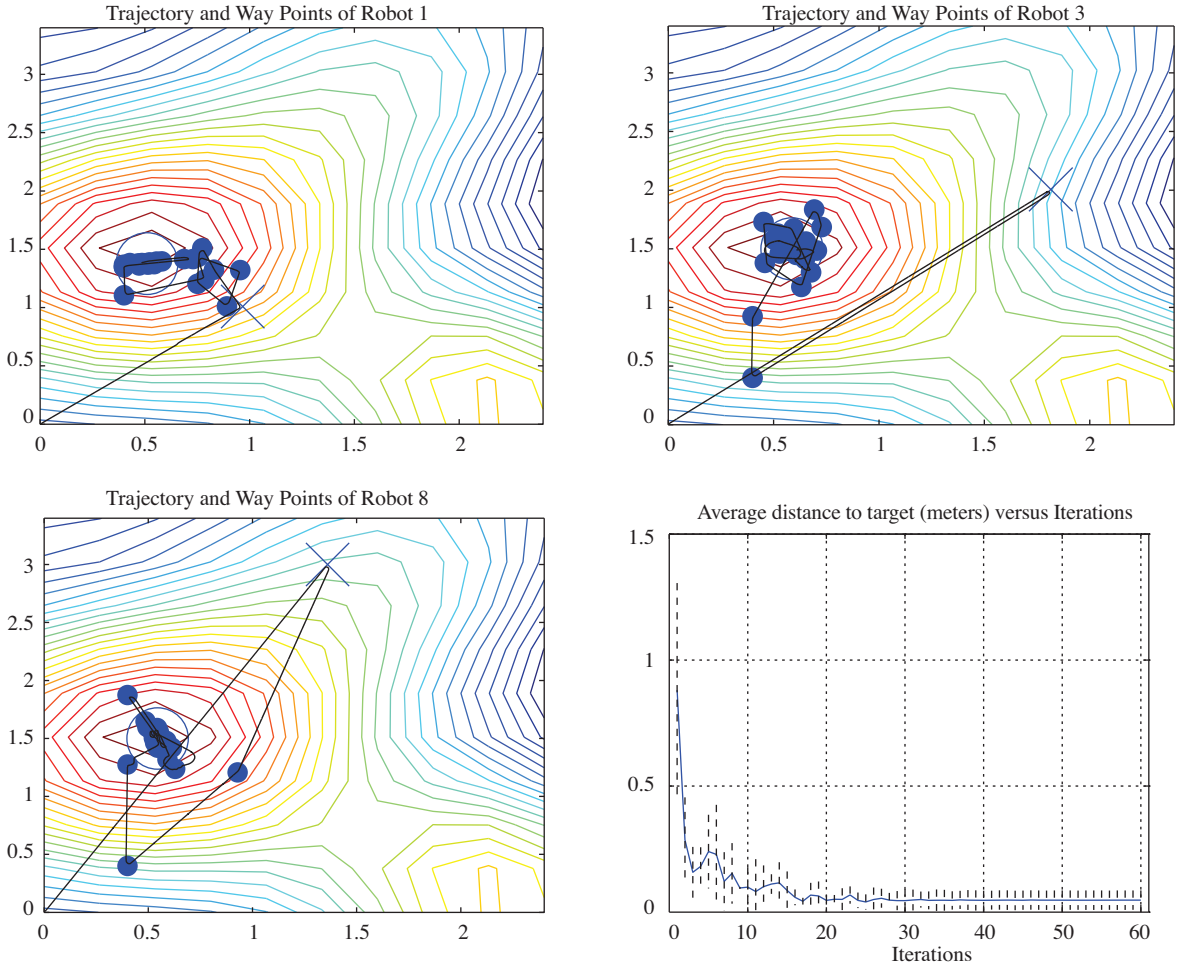

Figure 7. Trajectories and way points of the KheperaIII robots and the average distance to the maximum.

sample the environment. Moreover, consider a real application in which the robots are equipped with chemical sensors and move in an environment with a real chemical gas.[2] The data obtained from such sensors is usually very noisy and algorithms which rely directly on gradient information may quickly fail.

We have not tested the performance of the other variants of the PSO algorithm or other direct search methods for this application, since the objective here is not to compare their performances. However, one can easily deduce that under adverse effects such as failures, the classical synchronous PSO with static/fixed neighborhood topology will probably fail while the proposed asynchronous PSO with dynamic neighborhood will continue operation with the remaining (unfailed) robots. This is because in the standard synchronous PSO implementation the robots will wait for all their neighbors to finish their respective iterations before proceeding to the next step (and they will get stalled and wait indefinitely for information from a failed neighbor), whereas in the PSO algorithm discussed here since the neighbors are not fixed and there is no need for synchronization the robots will continue their search.

[2]In fact, the work on performing search with real robots equipped with gas sensing hardware in an environment with real chemical (ethanol) gas continues in our laboratory.

5. Concluding Remarks

In this study Asynchronous Particle Swarm Optimization based robotic search algorithm is tested in simulation environment and implemented with real robots. Due to the inherent asynchronous operation of multi-robot systems, limited communication ranges as well as possible permanent or temporary agent or communication failures the proposed algorithm is more suitable for multi-robot search applications compared to the other PSO variants. The obtained simulation and implementation results show that the proposed PSO variant performs satisfactory in a experimentally obtained smoothed virtual gas profile. Future work can focus on implementing the algorithm in a and possibly dynamically changing environments with gas and using robots equipped with a real chemical gas sensing hardware. Efforts in this direction continue.

References

[1] E. Sahin, "Swarm robotics: From sources of inspiration to domains of application," in *Swarm Robotics: State-of-the-art Survey*, ser. Lecture Notes in Computer Science (LNCS 3342), E. Sahin and W. Spears, Eds. Berlin Heidelberg: Springer-Verlag, 2005, pp. 10–20.

[2] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Sixth International Symposium on Micromachine and Human Science*, 1995, pp. 39–43.

[3] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[4] S. Doctor, G. Venayagamoorthy, and A. Gudise, "Optimal pso for collective robotic search applications," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC-2004)*, vol. 2, 2004, pp. 1390–1395.

[5] J. Hereford, "A distributed particle swarm algorithm for swarm robotic applications," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC-2006)*, vol. 2, 2006, pp. 1678–1685.

[6] J. Hereford, M. Siebold, and S. Nichols, "Using the particle swarm optimization algorithm for robotic search applications," in *Proceedings of IEEE Symposium on Swarm Intelligence (SIS-2007)*, 2007, pp. 53–59.

[7] J. Pugh and A. Martinoli, "Inspiring and modelling multi-robot search with particle swarm optimization," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC-2002)*, vol. 2, 2002, pp. 1666–1670.

[8] ——, "Distributed adaptation in multi-robot search using particle swarm optimization," in *From Animals to Animats 10, Proceeding of the 10th International Conference on Simulation of Adaptive Behavior, SAB 2008*, ser. Lecture Notes in Artificial Intelligence (LNAI 5040), M. Asada, J. C. T. Hallam, J.-A. Meyer, and J. Tani, Eds. Berlin Heidelberg: Springer, 2008, pp. 393–402.

[9] L. Marques, U. Nunes, and A. de Almedia, "Particle swarm-based olfactory guided search," *Autonomous Robots*, vol. 20, no. 3, pp. 277–287, May 2006.

[10] K. Derr and M. Manic, "Multi-robot, multi-target particle swarm optimization search in noisy wireless environments," in *2nd International Conference on Human System Interaction*, Catania, Italy, May 2009, pp. 81–86.

[11] V. Gazi, "Particle swarm optimization with dynamic neighborhood," in *IEEE Signal Processing and Communication Applications Conference (SIU2007)*, Eskişehir, Turkey, June 2007, (in Turkish).

[12] S. B. Akat and V. Gazi, "Particle swarm optimization with dynamic neighborhood topology: Three neighborhood strategies and preliminary results," in *IEEE Swarm Intelligence Symposium (SIS-2008)*, St. Louis, Missouri, September 2008.

[13] V. Gazi, "Asynchronous particle swarm optimization," in *IEEE Signal Processing and Communication Applications Conference (SIU2007)*, Eskişehir, Turkey, June 2007, (in Turkish).

[14] S. B. Akat and V. Gazi, "Decentralized asynchronous particle swarm optimization," in *IEEE Swarm Intelligence Symposium (SIS-2008)*, St. Louis, Missouri, September 2008.

[15] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with local search," in *IEEE Congress on Evolutionary Computation*, 1, Ed., Edinburgh, Scotland, September 2005, pp. 522–528.

[16] J. F. Schutte, J. A. Reinbolt, B. J. Fregly, R. T. Haftka, and A. T. George, "Parallel global optimization with the particle swarm algorithm," *International Journal for Numerical Methods in Engineering*, vol. 61, pp. 2296–2315, 2004.

[17] B. Koh, A. George, R. Haftka, and B. Fregly, "Parallel asynchronous particle swarm optimization," *International Journal for Numerical Methods in Engineering*, vol. 67, pp. 578–595, 2006.

[18] M. Clerc and J. Kennedy, "The particle swarm—explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, February 2002.

[19] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the International Conference on Advanced Robotics*, Coimbra, Portugal, June-July 2003, pp. 317–323.