Balancing Exploration and Exploitation in Particle Swarm Optimization on Search Tasking

Article *in* Research Journal of Applied Sciences, Engineering and Technology · September 2014

DOI: 10.19026/rjaset.8.1117

3 authors, including:

Bahareh Nakisa
Queensland University of Technology
29 PUBLICATIONS **599** CITATIONS

SEE PROFILE

Naim Rastgoo
Queensland University of Technology
26 PUBLICATIONS **599** CITATIONS

SEE PROFILE

Research Journal of Applied Sciences, Engineering and Technology 8(12): 1429-1434, 2014
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: May 31, 2014 Accepted: June 20, 2014 Published: September 25, 2014

Balancing Exploration and Exploitation in Particle Swarm Optimization on Search Tasking

Bahareh Nakisa, Mohammad Naim Rastgoo and Md. Jan Norodin
Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia 43600 UKM Bangi, Selangor, Malaysia

Abstract: In this study we present a combinatorial optimization method based on particle swarm optimization and local search algorithm on the multi-robot search system. Under this method, in order to create a balance between exploration and exploitation and guarantee the global convergence, at each iteration step if the distance between target and the robot become less than specific measure then a local search algorithm is performed. The local search encourages the particle to explore the local region beyond to reach the target in lesser search time. Experimental results obtained in a simulated environment show that biological and sociological inspiration could be useful to meet the challenges of robotic applications that can be described as optimization problems.

Keywords: Exploration and exploitation, local search algorithm, particle swarm optimization, search tasking

INTRODUCTION

One of the appropriate tasks for mobile robotic is searching for one or more target in unknown environments. By equipping the robots with sensors, they can detect the target and exploring the search space. Robotics search is useful especially when the environment is hazardous or dangerous to humans. Examples include locating mines for de-mining (Acar *et al*., 2003; Gage, 1995), finding victims in a disaster area (Kantor *et al*., 2003) and planetary exploration (Landis, 2003). Search task by using multi-robot system is more beneficial than the single robots because it can be done in parallel and also reduce the search time taken to reach the target and improving the robustness against failure of single agents by redundancy as well as individual simplicity (Sahin, 2005). By adding more robots the robustness and scalability of the system increase. Although, the search task has been extensively studied but multi-robot system to search is not well explored and is developed recently.

One of the most important algorithms in this domain is Particle Swarm Optimization (PSO) that is introduced by Eberhart and Kennedy (1995). Since this time, this technique has been used to solve a variety of optimization problems. Recently, many researches have been inspired from Particle Swarm Optimization to investigate search strategies on the multi-robot systems. One of the first versions of PSO algorithm, which demonstrated has an acceptable performance in searching task on the multi-robot system, is introduced by Doctor *et al*. (2004). Hereford (2006) introduced a method namely Distributed PSO which eliminate the central robot to coordinate all the robots. This method demonstrated that is scalable for large number of robots. Hereford *et al*. (2007) proposed a new method that simplified the previous method and eliminate the global communication among the robots at each iteration and they do all the calculations locally until they found the better position. The result shows that this method is done successfully on the multi-robot search with three robots or more. Adaptations of PSO have been used for multi-robot odor search in several instances (Jatmiko *et al*., 2006; Marques *et al*., 2006).

While biologically motivated algorithms such as PSO are very effective in providing optimal solutions, they can be further improved by maintaining the correct balance between exploration and exploitation (Holland, 1992; Clerc and Kennedy, 2002). Because, in the first iterations the exploration is done in PSO while after some iterations the exploitation is preferred. Many researchers invented many methods to create an efficient balance between exploration and exploitation by hybridizing PSO algorithm.

Vesterstrøm *et al*. (2002) borrowed the idea of division of labor from research on insect swarm algorithms. In their hybrid particle swarm model, individuals in the swarm were assigned, after some number of iterations without improvement, to conduct local search. Local search was implemented by placing a particle at the population's global best position with a new random velocity vector. The division of labor modification was intended to improve performance on unimodal problems; this improvement was seen, though performance on multimodal functions was not significantly improved.

Corresponding Author: Bahareh Nakisa, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia 43600 UKM Bangi, Selangor, Malaysia

Another method that uses the hybridization is Optimization of a Profiled Corrugated Horn Antenna by Robinson *et al*. (2002). In this method, they hybridized the two algorithms PSO and GA by switching from one to the other after several hundred iterations. They realized that PSO to GA (PSO-GA) is the best and noted that the PSO outperformed both the GA and the GA-PSO hybrid, though the PSO-GA hybrid performed best of all. The result shows that PSO more effectively explores the search space for the best region, while GA is effective at finding the best point once the population has converged on a single region. In this study we propose a new method (APSO) to create an efficient balance between exploration and exploitation by hybridizing Basic PSO algorithm with A-Star algorithm (Hart *et al*., 1968).

To test the performance of the algorithm in the realistic system, large quantities of computational time may require. This limitation motivates the use of abstracted models, which uses approximations of details of the system, which have little impact on the targeted performance metrics. Therefore, to validate the effectiveness and usefulness of these algorithms, we developed a simulation environment for conducting simulation-based experiments in different scenarios and report our experimental results.

MATERIALS AND METHODS

Problem formulation: This algorithm models a set of potential solution as a swarm of particles searching in the search space. Each particle in the swarm begins with the randomized position (x_{ij}) and randomized velocity (v_{ij}) in the n-dimensional search space. x_{ij} represents the position of the particle index i in the j-dimension of the search space. Particles by flying through the search space optimized the candidate solutions. At each iteration step each particles update its velocity based on its past velocity ($v_{i,j}$ (t)), its past best position ($p_{best}\ (t)$) and the global past best position (g_{best} (t)). The next position of the particle is update based on the next velocity ($v_{i,j}\ (t+1)$) and the past position ($x_{i,j}\ (t)$). The equations of PSO that executed at each step of the algorithm are:

$$v_{i,j}(t+1) = \omega \times v_{i,j}(t) + c_1 \times r_1 \times \left(p_{best}(t) - x_{i,j}(t)\right) + c_2 \times r_2 \times (g_{best}(t) - x_{i,j}t)) \quad (1)$$

$$x_{i,j}(t+1) = v_{i,j}(t+1) + x_{i,j}(t) \quad (2)$$

where, the inertia weight ω (Shi and Eberhart, 1998) and acceleration constant c_1, c_2 are assumed to be 0.9…. 0.5 and 2 and 2, respectively and r_1, r_2 are the uniformly generated random number in the range of (0, 1). In the first iteration (t = 0), the first position of each robot is considered for $p_{best}\ (0)$ and $g_{best}\ (0)$ is the first position of the one of the robot that is selected randomly. The termination criteria are also need to be taken into account to get good solution in the acceptable time. In this study, the termination criteria is based on two condition:

- If one of the robots reaches the target
- The number of iterations exceeds maximum iterations, which are assumed to be 200 iterations

The PSO-inspired multi-robot search algorithm is motivated by using a one-to-one matching between particles in the PSO swarm and robots in the multi-robot system. We initially assume the robots by accessing to the map of the search space have complete knowledge about their location in the environment. There are some key differences between PSO in multi-robot search and PSO that require us to make some modifications to the algorithm.

Search space: The real space in this study is transformed into 2-dimensional search space that is divided into squares (units). Each unit in search space represents a square in the real world with a selected size (for the algorithm itself, the size does not play any important role). The center of each unit is considered as a point of Interest. It means that if the robot visits the center of the unit, the entire unit occupied by the robot. The 2-dimensional search space in this study contains a single target with the same size of a unit. The reason behind of discretization is to prevent the collision between the robots.

Robot: The geometrical shape of the robot is assumed as a circle that has the same size of a unit. The state of each robot in the search space is represented by six variables (x, y, v, θ_r, θ_c, t) that are the position of the robot in the 2-D dimensional search space, speed of the robot, head of the robot, the determined direction of the robot to move to the next position and time in that position respectively. In this study there are 8 different adjacent units around the current position of the robot and it is supposed that the robot can move to them.

Robot path planning: Due to the discretization of search space, path planning of a robot from its current position to the next goal position is also discretized and the robots must go through the center of the units. In this study the path planning is computed by A-Star algorithm that produces the optimal path from the start position to the goal position.

Velocity limitation: In this study we assume the velocity of robots is limited and they have limitation on how quickly can move and adjust their headings. The velocity of the robot is based on the discrete values and at each time step it can execute just one action. The

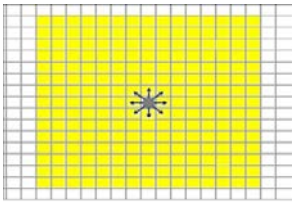

Fig. 1: The surrounding environment of the robot in its current position that is observed by the camera and 8 directions

velocity of the robots in this study is placed between [-V_{max} , V_{max}] where the v_{max} represent the maximum velocity of the robot along its direction and the $-v_{max}$ is the maximum velocity of the robot but in the reverse direction. If the velocity of the robot is placed out of this range we set this velocity as a Maximum velocity value in each side.

Fitness function: We assume each robot has a camera to capture the picture from the part of environment. This camera can only observe 7 units from the current position of the robot. When the robot uses the camera to find the target, if the target is placed in the range of view of the camera then evaluates the fitness function otherwise it returns zero. The fitness function in this study is as follows:

$$0 < fitness\ function = \frac{\sum_{i=1}^{n} po_i}{\sum_{j=1}^{m} p_j} < 1 \qquad (3)$$

where, $po = \{po_1, po_2, po_3 \dots po_n\}$ is a set of pixels of the target in the image captured by the camera and $p = \{p_1, p_2, p_3 \dots p_n\}$ is a set of pixels in the image captured by the camera. In this study, the robot is able to use their cameras in 8 different directions. Therefore, it has the ability to observe it's surrounding (can observe around 7 units around itself) by rotating its camera. When the robot stand in one unit we assume that the robot can rotates and takes pictures in 8 directions. Figure 1 shows the 8 directions of the robot in the current position and its surrounding environment that is observed by the camera.

Robot collision: Using the standard PSO particle displacement at each iteration, we will be unable to detect any collisions that might occur along the path. We therefore need to approximate the continuous movement of the robots by dividing the displacement into multiple steps and checking for collisions at each. In multi-robot system, robots and the target have some volume therefore they have to prevent to collide with each other or static obstacles. In this study we use the method that is introduced by Liu *et al*. (2012) to prevent robots from possible collisions. In this new method each robot generate its route independently and then checks the collision between them. There are separate paths for each robot from the initial position to the goal position. The aim of this method is to find the optimal path, which is the path with the lowest total cost. In this new method each robot replan their route as optimality as possible.

```
APSO Procedure
Do
        Initialization ();
        Calculate Fitness Function ();
        While (Stopping a criterion is not satisfied)
        Do
                If Fitness Function >0
                        A-Star ();
                Else
                        Calculate BPSO-velocity;
                Calculate New-Position ();
        End While
End
```

Fig. 2: Pseudo code of APSO algorithm

Hybridizing basic PSO with A-star: The proposed algorithm (APSO), which is based on the Basic PSO, is divided into 5 steps. The first step is initialization that places the robots in the search space randomly with the random initial velocity and directions. In this step the initial value of p_{best} , g_{best} is the initial position of the robot and the initial position of one of robots recursively.

In the second step, the camera of the robots take a picture from their surrounding search space of the robot current position and the fitness function of each robot is calculated based on the Eq. (3). The camera in this study has a limitation and it can just observe only 7 units from the current position of the robot. Therefore, if the target does not place in the range of view of the camera, the robot without any information from the target position should explore the search space. In this study, initially the robots do not have any information about the target position and just by applying their camera explore the environment and try to find the target.

The value of p_{best} and g_{best} is updated in the second step. If the fitness function value becomes better than any value found thus far, then the value of p_{best} is updated. The particle with the closest position to the goalobtains the highest value in the fitness function and g_{best} is updated. Then in the third step, there are two strategies to calculate the next velocity of each robot. In the next steps, if the fitness function is more than zero it means that the target is placed in the range of view of the camera and the robot is placed near the target and then A-Star is executed to do the Local search, otherwise the Basic PSO is executed to explore the search space. Figure 2 shows the Pseudo code of APSO algorithm.

In this study, once the fitness function become greater than zero then the robots by applying A-star

algorithm move toward the target directly. Otherwise, by using Basic PSO formulas Eq. (1) and (2), the robot moves to the next position.

In the A-Star algorithm, the robot starts from the current position and continues until reaching the determined position (Look ahead, which is equal to 1 in this study). It means that the robot by the A-star algorithm can move only one step and go to the adjacent units by specific direction. When the robot camera rotates, the robot can evaluate the fitness function for all 8 directions. Then, the A-star algorithm selects the largest F-value that belongs to a specific direction and the robot moves toward the adjacent cell along this specific direction. The F-values for these directions are calculated using the following formula:

$$f(n) = h(n) + g(n) \quad (4)$$

According to Eq. (4), the h (n) is the cost-to-go, which is assumed the fitness function value of the robot's current position in a specific direction. g (n) is the cost-thus-far, which is the cost from its current position to the next position. Due to the look ahead in this study is one, the g (n) is also equal to one. There are two lists in this algorithm; namely, Open and Close list. All the acceptable directions of the robot, which have a specific fitness function value, are stored in the Open list and then sorted based on the Max-heap. When each direction is added to the Open list, the list is reordered based on the biggest F-value and therefore, the top of the list refers to the biggest F-value. The selected direction with the biggest F-value pops up from the Open list and is put in the Close list. Then, the algorithm selects a state from the neighbor of the current state of the robot and guides the robot to move to the state with the best fitness function value. Figure 3 shows the Pseudo code of the A-Star algorithm.

In the algorithm, zero is set for all states cost-thus-far (g (*s*)) to the goal firstly. In the next step, the current location of the robot is put in the Open list. Then, the algorithm executes several instructions within specific time in a loop. In each time in the loop, the algorithm deletes maximum F-value state from Open list and stores it in Close list. In this step, the algorithm obtains the possible actions for robot to move toward its neighbors (children) states from deleted states. After that, if all the possible child states are not exist in the Open list then they will be stored in the Open List and also their parent states stored in the other list (tree) is stored.

RESULTS AND DISCUSSION

Simulation condition: We simulated and tested APSO and Basic PSO in four different initial robots position and a target position. Figure 3 shows the search space and the target position. In this study the search time was selected as a measurement to compare the performance

```
Procedure A-star
Do
for each s∈S
        Do
g(s): = 0;

        End
        g(start): = 0;
        Open =Close: = ϕ;
        Insert start into Open;
        expansions: = 0;
While expansion < Lookahead
        Do
              expansions:= expansion +1;
Delete a state s with the largest f-value (g(s)+h(s)) from Open;
Close: =Close ∪ {s};
               For each a ∈ A(s)
               Do
g(succ(s,a)) := g(s) + constant; tree (succ (s,a));
Ifsucc(s,a) is not in Open then insert into Open;
                End
         End
End
```

Fig. 3: Pseudo code of A-star algorithm

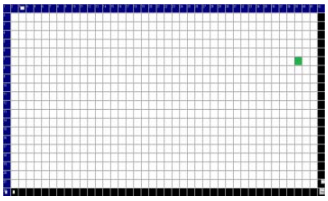

Fig. 4: Map of simulation search space and the 4 different target point locations

of algorithms. In each initial robots position, each algorithm performed 100 test cases. To calculate the next velocity, there are two random values (r_1, r_2) that are randomly selected in each test case.

The overall performance APSO was evaluated (i.e., the number of iterations that were found) and then compared with Basic PSO. Figure 4 compares the performance of APSO with Basic PSO in four different initial robots positions. The figures show the search times (number of iterations passed) for both APSO and Basic PSO algorithms. In this Study, the search space is assumed to be bounded with borders and thus, the robots cannot go outside the search space. Due to the condition approximation of the actual robot searching, the search space in our simulation is a hard border. It is assumed that when the next positions of the robots were set out of the search space, it should reverse and be placed inside the search space. For the simulation results the inertia coefficient, ω, was set to 0.9…. 0.5 and the both coefficient c_1, c_2 were set to 2. We set an initial value for v (velocity) for each robot to simulate the behavior of a physical robot. Only three robots were used; therefore, the l_{best} topology was identical to the g_{best} topology. In this study, we adapted the PSO algorithm to the multi-robot search system; therefore,

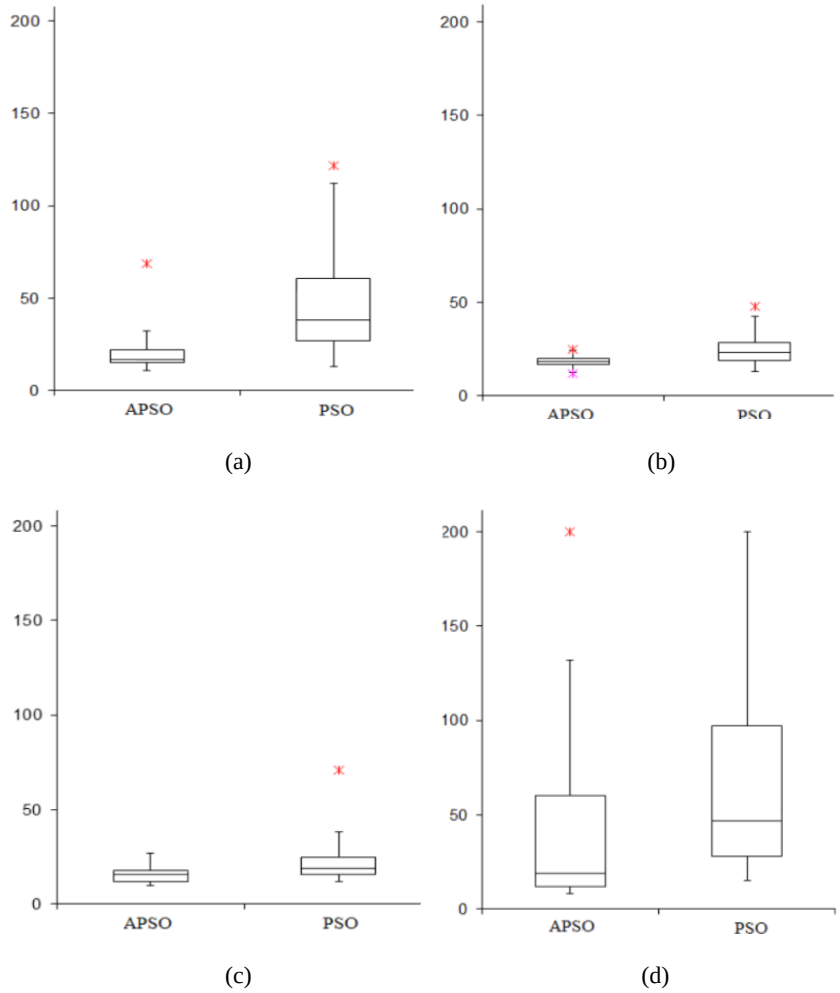

Fig. 5: The result of the search times of APSO and basic PSO algorithms with different initial robots position, (a) initial robots position 1, (b) initial robots position 2, (c) initial robots position 3, (d) initial robots position 4

unlike most of the PSO studies that have tracked the function value, our simulation searched the target function. The simulation stopped when the robot reach the target or when the maximum number of iterations (200 iterations) occurred.

Simulation result: To evaluate the effectiveness of the APSO and Basic PSO we made several simulation runs. We used the combination of an initial target positions and four initial robot positions to made the worst case in each test case. In this study, to compare the performance of algorithms the search time was considered as a measurement. In each test case, each algorithm performed 100 runs. To calculate the next velocity, there are two random values (r_1, r_2) that are randomly selected in each test case. The overall performance of *APSO* was evaluated (i.e., the number of iterations that were found) and then compared with *Basic PSO*. Figure 5 compares the performance of *APSO* with *Basic PSO* in 4 different initial robots position and shows the search times (number of iterations passed) for both APSO and Basic PSO algorithms.

The results demonstrate that APSO need lesser search time to reach the target than the Basic PSO. In the first initial robots positions (Fig. 5a), APSO algorithm can reach the target between 20-30 iterations whereas the search time of Basic PSO is between 20-110 iterations. The number of iteration in test case 2 (Fig. 5b) for both algorithm is under 50 iteration while APSO can find and reach the target in lesser time (around 25-30 iterations). As can be seen from the Fig. 5d, the search time of both algorithm is more than the other test cases and this is because of more distance between the target and initial robots positions. Although the search time in both algorithm is more but APSO has a better performance compared with Basic PSO.

CONCLUSION

We developed and tested a biologically inspired search strategy for multi-robot. This technique is a

hybridization of basic PSO with A* algorithm that is called APSO. One of the problems of Basic PSO on the multi-robot search system that APSO can solve is creating a balance between exploration and exploitation. In some cases the robot is near to the target and the fitness function value is high enough but the Basic PSO algorithm guide the robot to move to the positions which causing move away from the target that increase the search time. To decrease the search time and increases the global convergence, the A* algorithm (Local search algorithm) is used in this study to guides the robot to moves toward the target directly. When the robot see the target by applying A* algorithm can reaches the target in lesser search time instead of usinf Basic PSO formula. The results on the simulation of Multi-robot search system show that APSO has a better performance compared with Basic PSO algorithm and can reach the target in lesser search time.

REFERENCES

Acar, E.U., H. Choset, Z. Yangang and M. Schervish, 2003. Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods. Int. J. Robot. Res., 22(7-8): 441-466.

Clerc, M. and J. Kennedy, 2002. The particle swarm-explosion, stability and convergence in a multi-dimensional complex space. IEEE T. Evolut. Comput., 6: 58-73.

Doctor, S., G.K. Venayagamoorthy and V.G. Gudise, 2004. Optimal PSO for collective robotic search applications. Proceeding of the Congress on Evolutionary Computation (CEC, 2004), 2: 1390-1395.

Eberhart, R. and J. Kennedy, 1995. A new optimizer using particle swarm theory. Proceeding of the 6th International Symposium on Micro Machine and Human Science (MHS'95).

Gage, D.W., 1995. Many-robot MCM search systems. Proceeding of the Autonomous Vehicles in Mine Countermeasures Symposium. Monterey, CA, pp: 4-7.

Hart, P.E., N.J. Nilsson and B. Raphael, 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE T. Syst. Sci. Cyb., 4(2): 100-107.

Hereford, J.M., 2006. A distributed particle swarm optimization algorithm for swarm robotic applications. Proceeding of the IEEE Congress on Evolutionary Computation (CEC, 2006).

Hereford, J., M. Siebold and S. Nichols, 2007. Using the particle swarm optimization algorithm for robotic search applications. Proceeding of IEEE Symposium on Swarm Intelligence (SIS, 2007), pp: 53-59.

Holland, J.H., 1992. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. The MIT Press, Cambridge.

Jatmiko, W., K. Sekiyama and T. Fukuda, 2006. A PSO-based mobile sensor network for odor source localization in dynamic environment: Theory, simulation and measurement. Proceeding of the IEEE Congress on Evolutionary Computation. Vancouver, BC, Canada, July 16-21, pp: 1036-1043.

Kantor, G., S. Singh, R. Peterson, D. Rus, A. Das, V. Kumar, G. Pereira and J. Spletzer, 2003. Distributed search and rescue with robot and sensor teams. Proceeding of the 4th International Conference on Field and Service Robotics, Japan.

Landis, G.A., 2003. Robots and humans: Synergy in planetary exploration. Acta Astronaut., 55(12): 985-990.

Liu, F., A. Narayanan and Q. Bai, 2012. Effective methods for generating collision free paths for multiple robots based on collision type. Proceeding of the 11th International Conference on Autonomous Agents and Multi-agent Systems, Vol. 3.

Marques, L., U. Nunes and A.T. de Almeida, 2006. Particle swarm based OL-factory guided search. Auton. Robot., 20: 277-287.

Robinson, J., S. Sinton and Y. Rahmat-Samii, 2002. Particle swarm, genetic algorithm and their hybrids: Optimization of a profiled corrugated horn antenna. Proceeding of the IEEE International Symposium in Antennas and Propagation Society, pp: 314-317.

Sahin, E., 2005. Swarm robotics: From sources of inspiration to domains of application. In: Şahin, E. and W. Spears (Eds.), Swarm Robotics Workshop: State-of-the-Art Survey. Berlin, Germany, Lect. Notes Comput. Sci., 3342: 10-20.

Shi, Y. and R. Eberhart, 1998. A modified particle swarm optimizer. Proceeding of the IEEE International Conference on Evolutionary Computation and IEEE World Congress on Computational Intelligence.

Vesterstroøm, J.S., J. Riget and T. Krink, 2002. Division of labor in particle swarm optimization. Proceeding of the IEEE Congress on Evolutionary Computation (CEC, 2002). Honolulu, HI, Piscataway, pp: 1570-1575.