FACTA UNIVERSITATIS
Series:**Automatic Control and Robotics** Vol. 14, N^o 2, 2015, pp. 77- 86

COMPUTATION OF THE SHORTEST DISTANCE BETWEEN TWO PARAMETRIC DEFINED OBJECTS BY PARTICLE SWARM OPTIMIZATION

UDC (681.58:004.89):681.513.3

Predrag M. Rajković, Emina P. Petrović, Vlastimir D. Nikolić

University of Niš, Faculty of Mechanical Engineering, Niš, Republic of Serbia

Abstract. *The distance computation between objects is an essential component of robot motion planning and controlling the robot to avoid its surrounding obstacles. Distance is used as a measure of how far a robot is from colliding with an obstacle. In this paper a Particle Swarm Optimization algorithm (PSO) for solving the problem of the distance computation between convex objects is presented. Convergence analysis of the suggested method was done via difference equation.*

Key words: *Particle swarm optimization, minimum distance computation, parametric surfaces,convergence analysis.*

1. INTRODUCTION

One of the fundamental problems in robotics represents the distance computation between objects. It presents a necessary component of robot motion planning and controlling the robot to avoid its surrounding obstacles. The obstacles can be polyhedral objects or quadratic surfaces that include spherical and cylindrical surfaces or more general, surfaces such as splines [1]. For all developed research for path planning it is necessary to possesses at least the lowest ability to detect if collision has occurred [2]. Collision detection technique is related to the distance computation between objects: two objects are in collision if and only if the distance between the objects is zero [3]. One of these objects, for which we examine whether or not is in collision, is the robot. The set of all the obstacles which surround the robot in its environment represents the other object Representing these two objects by mathematical models we can find a point on each object and compute the distance between the points so that the distance between the points is minimized [3].

A lot of research work on the minimum distance computation on the convex objects has been done in recent years.

Received January 22, 2015
Corresponding author: Emina Petrović
Faculty of Mechanical Engineering, Aleksandra Medvedeva 14, 18000 Niš, Republic of Serbia
E-mail: emina@masfak.ni.ac.rs

Chen et al. [4] in their work used an analytical technique to determine the minimum distance between two algebraic surfaces. An efficient algorithm for the minimum distance computation between non-convex objects using the sphere bounding volume hierarchy representation is described by Quinlan [3]. In his work, Lumelsky [5] calculated the minimum distance between three-dimensional segments. The algorithms for finding the distance between two convex polyhedral objects are described by Bobrow [6], Gilbert et al. [7], Cameron [8], and Lin [9]. In these algorithms the distance monotonically converges to a minimum. Iteratively found pair of points, one on each object, represents the required distance. However, the mechanism of these algorithms completely depends on the properties of the convex objects, and it appears difficult to extend them directly to the non-convex objects.

The computational efficiency of two-dimensional algorithms for polygons was given in [10]. Rabl et al. [11] used the class of surfaces with quadratic polynomial support functions in order to define bounding geometric primitives for the shortest distance computation. The common normals of two such surfaces have been computed by solving a single polynomial equation of degree six. Based on this observation, they formulate an algorithm for computing the shortest distance between enclosures of two moving or static objects by surfaces of this type.

In this research for solving the problem of computation of the shortest distance between convex objects is done by a Particle Swarm Optimization algorithm (PSO). In this paper, we only consider the case where the two surfaces do not intersect. The method is simple, easy to implement and there are fewer parameters that need to be adapted.

Also, in this research, we will expose a few trials of the convergence criteria researching for particle swarm algorithm and will compare them with experiments.

2. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization is a population based stochastic search algorithm that is the most recent development in the of category combinatorial meta-heuristic optimization. It was first introduced by Kennedy and Eberhart in 1995 [12], as a new heuristic method inspired by the social behavior exhibited by flocks of birds flying across an area looking for food.

In the basic particle swarm optimization, particle swarm consists of n particles, and the coordinates of each particle represent a possible solution called particles associated with position and velocity vector in D-dimensional space [13].

At each iteration, particle moves towards the optimum solution, through its present velocity, personal best solution obtained by themselves so far and global best solution obtained by all particles.

We represent the position of ith particle of the swarm by a D-dimensional vector $\boldsymbol{x}_i = (x_1, x_2, \dots, x_D)$.The velocity (position change per generation) of the particle $\boldsymbol{x}_i$ can be represented by another D-dimensional vector $\boldsymbol{v}_i = (v_1, v_2, \dots, v_D)$. The best position previously visited by the ith particle is denoted as $\boldsymbol{b}_i = (b_1, b_2, \dots, b_D)$. If the topology is defined so that all particles are assumed to be neighbors and g as the index of the particle visited the best position in the swarm, then $\boldsymbol{p}_g$ becomes the best solution found so far, and the velocity of the particle and its new position will be determined according to the following two equations:

$$\mathbf{v}_i^{(m+1)} = w\mathbf{v}_i^{(m)} + c_1 r_1 (\mathbf{b}_i^{(m)} - \mathbf{x}_i^{(m)}) + c_2 r_2 (\mathbf{p}_g^{(m)} - \mathbf{x}_i^{(m)}), \tag{1}$$

$$\mathbf{x}_i^{(m+1)} = \mathbf{x}_i^{(m)} + \mathbf{v}_i^{(m+1)}, \tag{2}$$

r_1 and r_2 are random variables in the range [0,1]; c_1 and c_2 are acceleration coefficients regulating the relative velocity toward global and local best [14-16].

Some authors use notation:

$$c = c_1 + c_2, \qquad \varphi_k = c_k r_k\,(k = 1,2), \qquad \varphi = \varphi_1 + \varphi_2 .$$

The PSO flowchart can be described as follows:

1. Generate the initial particles by randomly generating the position and velocity for each particle.
2. Evaluate each particle's fitness.
3. For each particle, if its fitness is smaller than its previous best (***b****ᵢ*), update ***b****ᵢ* .
4. For each particle, if its fitness is smaller than the best one ($\boldsymbol{p}_g$) of all particles, update $\boldsymbol{p}_g$.
5. For each particle generate a new particle *t* according to the formula (1) and (2).
6. If the stop criterion is satisfied, then stop, else go to Step 3.

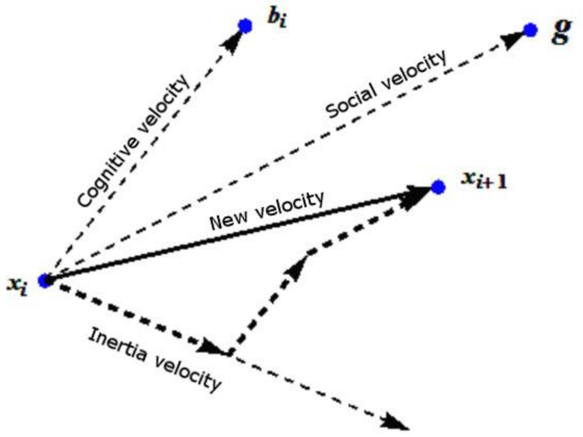


Fig. 1 Velocity and position update for a particle in a two-dimensional search space.

3. Convergence Analysis of PSO Algorithm

Here, we will expose a few trials in researching the convergence criteria for particle swarm algorithm and compare them with experiments.Van den Bergh [17] proved that the PSO is not a global search algorithm, even not a local one.

The analysis of the particle swarm optimization done by Clerc and Kennedy [18] lead to a generalized model of the algorithm, containing a set of coefficients to control the system's convergence tendencies. Simplification of the system was done by stripping the algorithm down to almost a simple form.

The initial investigation was simplified by looking at the behavior of a particle whose velocity is adjusted by only one term:

$$\mathbf{v}_i^{(m+1)} = \mathbf{v}_i^{(m)} + \varphi_1(\mathbf{b}_i^{(m)} - \mathbf{x}_i^{(m)}) + \varphi_2(\mathbf{p}_g^{(m)} - \mathbf{x}_i^{(m)}), \tag{3}$$

i.e.,

$$\mathbf{v}_i^{(m+1)} = \mathbf{v}_i^{(m)} + \varphi\left(\frac{\varphi_1}{\varphi}(\mathbf{b}_i^{(m)} - \mathbf{x}_i^{(m)}) + \frac{\varphi_2}{\varphi}(\mathbf{p}_g^{(m)} - \mathbf{x}_i^{(m)})\right). \tag{4}$$

The formula can be shortened by being redefined as follows:

$$\mathbf{v}_i^{(m+1)} = \mathbf{v}_i^{(m)} + \varphi\left((\frac{\varphi_1\mathbf{b}_i^{(m)} + \varphi_2\mathbf{p}_g^{(m)}}{\varphi} - \mathbf{x}_i^{(m)})\right). \tag{5}$$

Denoted by:

$$\mathbf{P}_i^{(m)} = \frac{\varphi_1\mathbf{b}_i^{(m)} + \varphi_2\mathbf{p}_g^{(m)}}{\varphi}, \tag{6}$$

We can write:

$$\mathbf{v}_i^{(m+1)} = \mathbf{v}_i^{(m)} + \varphi(\mathbf{P}_i^{(m)} - \mathbf{x}_i^{(m)}). \tag{7}$$

The system can be simplified even further by considering a one-dimensional problem space and again further by reducing the population to one particle.

When the particle swarm operates on an optimization problem, the value of $\mathbf{P}_i^{(m)}$ and φ are constantly updated, as the system evolves towards the optimum. In order to further simplify the system and make it understandable, Clerc sets them to two constant values in the following analysis:

$$\mathbf{P}_i^{(m)} \equiv \mathbf{P} \quad (\forall i, m), \qquad \varphi(c_1, r_1, c_2, r_2,) = \varphi = const.$$

Since φ is defined as a random number between zero and a constant upper limit, it removed the stochastic component.

F. van der Bergh [17] noticed that there is no interaction between the different dimensions; they are independent of each other. So the problem can be simplified from multi-dimensional space into one dimension space and dimensional indices can be dropped.

S. Gao and J.Y. Yang [19] in their research, supposed that particles are independent of each other and the swarm's global best position and the best position attained by particle self are unchanged, i.e. that there exists some positive integer number m_0, so that:

$$\mathbf{b}^{(m)} = \mathbf{b}, \ \mathbf{p}_g^{(m)} = \mathbf{g}, \ (m = m_0 + 1, m_0 + 2, \ldots).$$

Then, the swarm method can be written in the form of a difference equation:

$$\mathbf{x}^{(m+2)} - (1 + \omega - \varphi_1 - \varphi_2)\mathbf{x}^{(m+1)} + \omega\mathbf{x}^{(m)} = \varphi_1\mathbf{b} + \varphi_2\mathbf{g}, \tag{8}$$

Hence, the convergence zone was established:

$$-1<\omega<1, \ c<2(\omega+1).$$

*Example 1.*We have considered the Rosenbrock function:

$$f(x) = \sum_{i=1}^{D-1}\left[100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2\right], \tag{9}$$

which has minimum $f_{min} = 0$ in $\mathbf{x}^* = (1,1,\ldots,1)$.

The initial values were random numbers from the interval [-2, 2].We accepted that the method is converging to the optimal point if , after m=100 iterations, it is fulfilled:

$$||\mathbf{x}* - \mathbf{g}|| < 0.1 \wedge |f(\mathbf{x}^*) - f(\mathbf{x})| < 0.01 .$$

In Fig. 2, we showed convergence for different values of parameters

$$\omega = 0.1(0.1),\ c = 1(0.1)10,\ c_1 = c_2 = \frac{c}{2} .$$

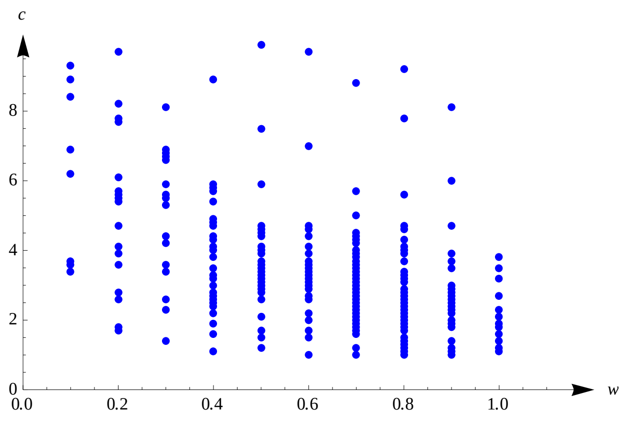


Fig. 2 The convergence for different values of parameters w and c.

In Fig. 3, we see convergence for fixed ω=0.8 and different values of $c_1, c_2 \in [1,5]$.

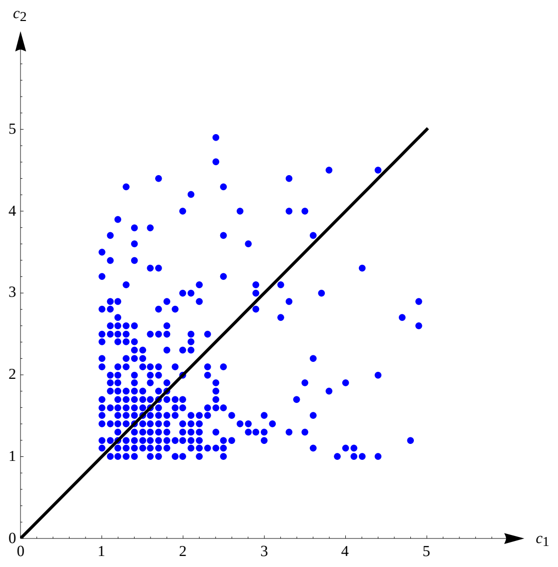


Fig. 3 The convergence for fixed value of parameter ω=0.8 and different values of parameters $c_1, c_2 \in [1,5]$.

4. The Shortest Distance between two Objects by Pso

In this research we use an algorithm based on Particle swarm optimization for the shortest distance computation between two of convex objects presented by parametric surface [19]. Also, the performance of this algorithm will be demonstrated by several examples. Let us consider two objects in the 3D space $A \subset R^3$ and $B \subset R^3$. The Euclidean distance between the object A and B is defined by their closest points, i.e.:

$$d(A,B) = \min\{|P-Q| : P \in A, Q \in B\}. \tag{10}$$

If *A* and *B* intersect, d_{min} is equal to zero. In this paper, we assume that *A* and *B* do not intersect.

Since in this paper we use parametric representation of objects, they are given by

$$(X_A, Y_A, Z_A)^T = (X_A(u_A, v_A), Y_A(u_A, v_A), Z_A(u_A, v_A))^T, \tag{11}$$

$$(X_B, Y_B, Z_B)^T = (X_B(u_B, v_B), Y_B(u_B, v_B), Z_B(u_B, v_B))^T, \tag{12}$$

where u_A, v_A, u_B, v_B are parameters, then we have the equation system:

$$\begin{cases} f_A = (X_A(u_A, v_A), Y_A(u_A, v_A), Z_A(u_A, v_A)) \\ f_B = (X_B(u_B, v_B), Y_B(u_B, v_B), Z_B(u_B, v_B)). \end{cases} \tag{13}$$

The main idea of the algorithm presented in this paper is that when two convex objects are placed in a 3D space, the algorithm needs to find a pair of points, one on each object, so that the distance between the points is less than or equal to the distance between any other pair.

Example 2.

To demonstrate the performance of the purposed algorithm we considered two spheres:

$$\begin{cases} S_1 = (C_1(0,0,0), r_1 = 1) \\ S_2 = (C_2(4,4,4), r_2 = 1). \end{cases} \tag{14}$$

It is exactly known that the nearest points between those spheres are

$$A = \left(\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3} \right) \text{ and } B = \left(4 - \frac{\sqrt{3}}{3{,}4} - \frac{\sqrt{3}}{3{,}4} - \frac{\sqrt{3}}{3} \right).$$

For which we know their exact minimum distance

$$d_{\min} = \left(4 - 2\frac{\sqrt{3}}{3} \right)\sqrt{3} = 4.9282.$$

We tested our algorithm and compared the obtained results. In the implemented algorithm after 100 iterations the obtained results are:

1. $d_{min}=4.93061$.

2. Coordinate of point on first sphere are:

$(x_A, y_A, z_A) = (0.529669, 0.580306, 0.618624)$.

3. Coordinate of point on second sphere are:

$(x_B, y_B, z_B) = (3.41942, 3.42806, 3.42051)$

The error between the calculated distance and the distance obtained by PSO is -0.00241, which is quite acceptable for practical uses in robotics.

In the picture below (Fig. 4), it is showed how PSO positioned particles on the surface of both spheres in order to find a pair of particle, one on each object which has the smallest distance.

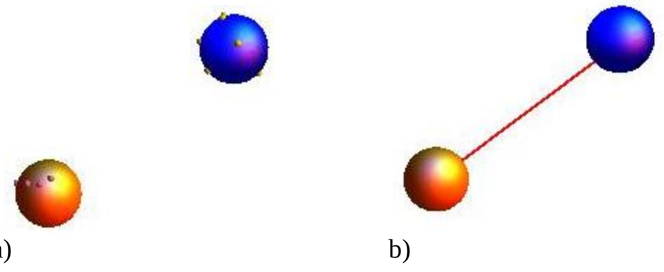


Fig. 4 a) The PSO positioned particles on the surface of two spheres, b) the minimum distance between two spheres obtained by PSO.

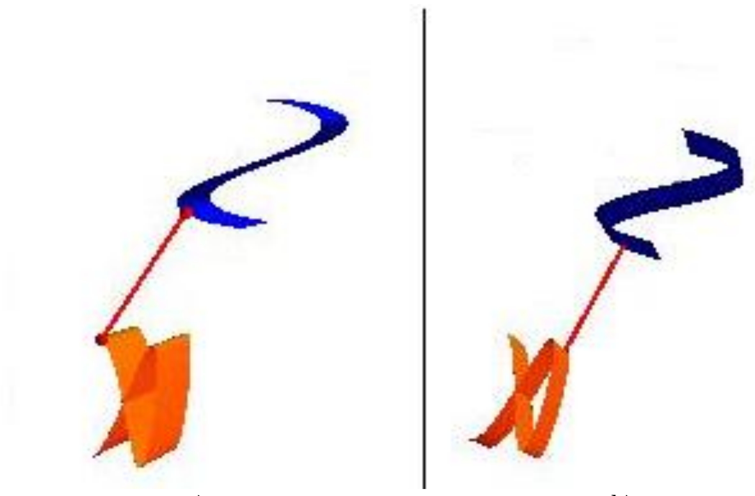
5. Results

In order to verify the results in the picture below are presented some examples where PSO found the minimum distance for different objects and compared them with the results provided by Newton method. The obtained results are presented in Table 1.

From pictures and data from Table 1, it can be seen that the distances calculated by both methods are similar in some cases, especially when the objects are not too complex (e.g. Fig. 5). But in cases when we have complex objects, the PSO gives better results than Newton method.

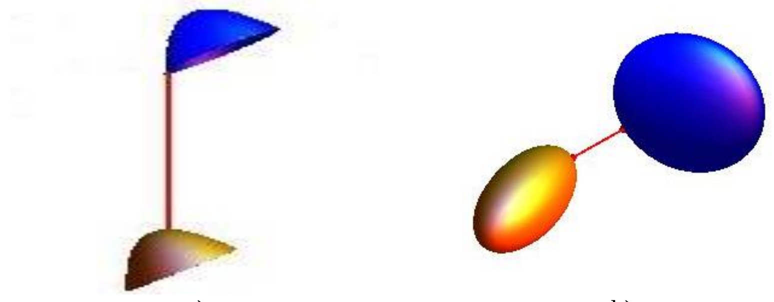
Table 1 The distance obtained by Newton method and PSO

Fig. No	Newton method	PSO
6	2.41642	2.145369
7a	3.52632	3.52634
7b	2.053835	2.05979
8a	1.382541	1.392541
8b	2.42259	2.42865



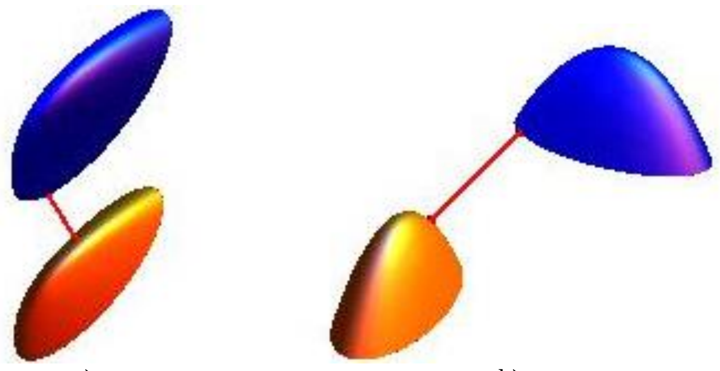
a) b)

Fig. 5 The minimum distance between two complex objects obtained:
a) by Newton method, b) by PSO.



a) b)

Fig. 6 The minimum distance obtained by PSO between: a) two cones, b) two ellipsoids.



a) b)

Fig. 7 The minimum distance obtained by PSO between:
a) two parallel cones, b) two normal positioned cones.

6. CONCLUSION

The minimum distance computation problem usually involves non-linear equation systems. It is possible, but to complicated and difficult, to compute all the roots for the corresponding general non-linear equation systems, but it is not necessary to do so.

In this paper, the utilization of the Particle swarm optimization method for the minimum distance computation between two objects which are represented by parametric surfaces, is presented Using this algorithm is simple, easy to implement and there are fewer parameters needed to be adapted

Because of the nature of the algorithm it is difficult to make general statements about its performance, as it relies on many factors that depend on the application for which the distance algorithm is used.

This algorithm maybe not always give exact results as a specialized algorithm, e.g. Newton and gradient method in application where it is possible to use it, but the obtained results are quite acceptable.

Another big advantage of Particle Swarm optimization algorithm is its possibility to be used in a wide range of applications.

The methodology described here can be used in distance calculation problems, real-time path planning and other robotics problem. This application is to plan to use in real-time path planning in order to avoid obstacles during human tracking. In addition we are working to improve the efficiency of Particle Swarm optimization method in order to improve the convergence and reducing errors.

REFERENCES

[1] M. S. Uddin, K. Yamazaki, "Study of distance computation between objects represented by discrete boundary model," International Journal of Graphics, vol. 1, no. 1, pp.29–43, 2010.

[2] Ch. L. Shih, J. Y. Liu, "Computing the minimum directed distances between convex polyhedra", Journal of Information Science and Engineering, vol. 15, pp.353–373, 1999.

[3] S. Quinlan, "Efficient distance computation between non-convex objects", in *Proceedings of IEEE International Conference on Robotics and Automation,* vol.4, pp. 3324–3329, 1994. [Online]. Available: http://dx.doi.org/10.1109/ROBOT.1994.351059.

[4] X. D. Chen, J. H. Yong, G. Q. Zheng, J. C. Paul, J. G. Sun, "Computing minimum distance between two algebraic surfaces", Computer-Aided Design, vol. 38, no. 10, pp.1053–1061, 2006. [Online]. Available: http://dx.doi.org/10.1016/j.cad.2006.04.012

[5] V. J. Lumelsky, "On fast computation of distance between line segments", Information Processing Letters, vol. 21, pp. 55–61, 1985. [Online]. Available: http://dx.doi.org/10.1016/0020-0190(85)90032-8

[6] J. E. Bobrow, "Optimal robot plant planning using the minimum time criterion", IEEE Journal of Robotics and Automation, vol. 4, no. 4, pp. 443–450, 1988. [Online]. Available: http://dx.doi.org/10.1109/56.811

[7] E. Gilbert, D. E. Johnson, S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space", IEEE Journal of Robotics and Automation, vol. 2, no. 4, pp.193–203, 1988. [Online]. Available: http://dx.doi.org/10.1109/56.2083

[8] S. Cameron, "Enhancing GJK: computing minimum and penetration distances between convex polyhedral", in *Proceeding of International Conference on Robotics and Automation*, Albuquerque, NM USA, pp. 3112–3117, 1997. [Online]. Available: http://dx.doi.org/10.1109/ROBOT.1997.606761.

[9] M. C. Lin, "Efficient Collision Detection for Animation and Robotics", Ph. D. Dissertation, University of California Berkeley, 1997.

[10] F. Chin, C. A. Wang, "Optimal algorithms for the intersection and minimum distance problems between planar polygons", IEEE Transactions on Computers, Vol. C-32, pp. 1203–1207, 1983. [Online]. Available: http://dx.doi.org/10.1109/TC.1983.1676186

[11] M. Rabl, B. Jüttler, "Fast distance computation using quadratically supported surfaces", Computational Kinematics, pp. 141–148, 2009.

[12] J. Kennedy, R. C. Eberhart, "Particle swarm optimization", in *Proceedings of IEEE International Conference on Neural Network*, pp. 1942–1948, 1995. [Online]. Available: http://dx.doi.org/10.1109/ICNN.1995.488968
[13] Q. Bai., "Analysis of particle swarm optimization algorithm", Computer and Information Science, vol. 3, no. 1, 2010. [Online]. Available: http://dx.doi.org/10.5539/cis.v3n1p180
[14] L. P. Zhang, H. J. Yu, S. X. Hu , "Optimal choice of parameters for particle swarm optimization", Journal of Zhejiang University SCIENCE, vol. 6, no. 6, pp. 528–534, 2005. [Online]. Available: http://link.springer.com/article/10.1631/jzus.2005.A0528
[15] L. Wang, C. Singh, " Stochastic combined heat and power dispatch based on multi-objective particle swarm optimization", *Power Engineering Society General Meeting, 2006. IEEE*, vol.30, pp. 226–234, 2008. [Online]. Available: http://dx.doi.org/10.1109/PES.2006.1709288.
[16] F. van den Bergh, "An analysis of particle swarm optimizers", Ph.D. dissertation, University Pretoria, Pretoria, South Africa, 2002.
[17] M. Clerc, J. Kennedy, "The particle swarm explosion, stability, and convergence in a multidimensional complex space", IEEE Transactions on Evolutionary Computation, vol. 4, no. 1, pp. 58–73, 2002. [Online]. Available: http://dx.doi.org/10.1109/4235.985692
[18] Sh. Gao, C. Cao, "Convergence analysis of particle swarm optimization algorithm", Advances in information Sciences and Service Sciences (AISS), vol. 4, no. 14, pp.25–32, 2012. [Online]. Available: http://dx.doi.org/doi:10.4156/AISS.vol4.issue14.4
[19] E. Petrović, P. Rajković, V. Nikolić, "Particle swarm optimization for computation the shortest distance between two objects", in *Proceedings of XII International Conference SAUM 2014*, Niš, Republic of Serbia, pp. 192–195, 2014.