Proceedings of the 7th WSEAS International Conference on Evolutionary Computing, Cavtat, Croatia, June 12-14, 2006 (pp45-52)

Complexity Analysis of Problem-Dimension Using PSO

BUTHAINAH S. AL-KAZEMI AND SAMI J. HABIB,
Department of Computer Engineering
Kuwait University
P.O. Box 5969 Safat, 13060 Kuwait
KUWAIT

Abstract: -This work analyzes the internal behavior of particle swarm optimization (PSO) algorithm when the complexity of the problem increased. The impact of number of dimensions for three well-known benchmark functions, DeJong, Rosenbrock and Rastrigin, were tested using PSO. A Problem-Specific Distance Function (PSDF) was defined to evaluate the fitness of individual solutions and test the diversity in neighboring individuals. The PSDF started with a large value, but converged to the optimum in few generations, irrespective of complexity of problem or objective benchmark function. The simulation illustrates that all parameters in any dimension behave in similar pattern and we can expect similar behavior for additional complexity in the problem.

Key-Words: - Evolutionary Computation, Genetic Algorithm, Particle Swarm Optimization, Distance Function, Parameter Prediction.

1. Introduction

The ability of Particle Swarm Optimization (PSO) algorithms, to find near optimal solutions to an optimization problem has been widely demonstrated[1]. However, one of the major concerns in applying these methods is the behavior of this algorithm when the number of variables or the dimension of the problem is increased. The numbers of unknowns have a significant impact on the algorithm's behavior, and therefore greatly affect the quality of the final solution found, as well as the time taken to find that solution [2] [3]. These results demonstrate that by understanding the behavior of the algorithm when using few unknowns, and the relationships between them, the computation required can be reduced significantly [4]. The objective of this paper is to demonstrate the effectiveness of the behavior analysis of PSO algorithm by varying complexity of problem-domain.

We observe that as the complexity of problem increases, there is not much change in the way an individual solution behaves. We tested and justified the behavior by comparing the computational performance of the PSO using a set of three mathematical benchmark objective functions namely, DeJong F1, Rosenbrock and Rastrigin [5]. We tried to analyze the behavior of individual elements and formulate a relation between consecutive increments in the number of unknowns from 2 to 6. Also we analyzed each dimension and observed the behavior of individuals in a particular dimension by varying the problem complexity.

We have used the distance function to evaluate the individuals according to their position from the best individual so far. The distance function (DF) is usually a measure of how far two solutions are from each other [6]. We have classified the distance functions into two groups: an algorithmic distance function ADF and a problem-specific distance function PSDF. In this paper, we use the PSDF.

This paper is organized in 5 sections. Section 2 describes the Particle Swarm Optimization Algorithm and defines the Distance Functions. In Section 3, the benchmark objective functions are discussed. Section 4 and 5 are a detailed explanation of our experiment and it includes PSDF value analysis for three benchmark functions uder study. In Section 6, we conclude our observations.

2. Particle Swarm Optimization and Distance Functions

2.1 PSO

Evolutionary Computational Algorithms (ECA) are gaining popularity especially when difficult optimization problems are studied in depth [7]. It complements the study of traditional computational systems [8]. There are several types of ECAs in use today. The four traditional algorithms are: the Genetic Algorithm (GA), Evolutionary Programming (EP), Evolution Strategies (ES), and Genetic Programming (GP). More recently, new ECAs have arisen, such as Particle Swarm Optimization (PSO), Ant Colony (ACO), and the Shuffled Frog Leaping Algorithm (SFLA).

The PSO is a population based stochastic searching technique, which was developed by Kennedy and Eberhart in 1995. It is one of the most popular methods for optimization of continuous nonlinear functions. One of the important merits of PSO Algorithms is their ability to tackle real-world problems, which are very hard or even unsolvable by traditional optimization techniques. According to [9], PSO is a biological motivated optimization paradigm which involves "social" interaction between individuals of a population.These individuals, also called particles, can move through a multidimensional search-space in order to find an optimum. Particles can change their movement directions and velocities according to the information which they themselves and the whole swarm have gathered so far from the fitness landscape. Each particle remembers its own best position it has encountered by then while all particles know about the overall best position which has been found by all particles of the swarm. From this information and each particle's current velocity vector new velocity vectors are calculated and the positions of the particles are updated. Different parameter settings can influence the behavior of the swarm, for example its ability to converge. PSO has been inspired by the social behavior of flock of birds or school of fishes. It uses population of individuals, where they are trying to reach the best solution [10].

In each generation, the swarm is flying with a certain velocity, which is used to change their positions. The changing of the position not only depends on the velocity but also on the current position of the particle itself in that instance and to the best individual's position so far. There is only one piece of food in the area being searched. All the birds don't know where the food is; however, the birds know how far the food is in each generation. Therefore, the effective strategy to find the food is to follow the bird that is the nearest to the food.

In PSO, each single solution is a bird in the search space and it is called a particle. All of particles have fitness and velocity values. The fitness value is evaluated by the fitness function to be optimized; moreover, the velocity value directs the flying of the particle through the problem space by following the current best particle. A modified version of PSO with inertia weights was introduced by Shi and Eberhart [4], as depicted in Figure 1.

```
Particle Swarm Optimization:
1 begin
2 t =0;
3 initialize particles P (t);
4 evaluate particles P (t);
5 while (termination conditions are unsatisfied)
6 begin
7   t = t + 1;
8   update weights
9   select pBest for each particle
10    select gBest from P (t-1);
11    calculate particle velocity P (t);
12    update particle position P (t);
13    evaluate particles P (t);
14  end
15 end
```

Fig 1: The basic structure of particle swarm optimization**.**

PSO algorithm executes two Equations 1-2 for updating the velocity and position of every particle in the population.

$$V[i] = w * V[i] + \left(C_1 \times rand() \times \left(pBest[i] - present[i]\right)\right) + \left(C_2 \times rand() \times \left(gBes[i] - present[i]\right)\right) \quad (1)$$

$$present[i] = present[i] + V[i] \quad (2)$$

In the equation 1 and equation 2, V[i] and present[i] represent the velocity and position of ith particle respectively. The pBest[i] represents the present best for the ith particle and gBest[i] represents the global best among the population so far. The function rand () generates random number uniformly. w is the inertia weight C1 and C2 are constants assigned value 2 most of the time.

2.2 Distance Functions

The distance functions (DF) are a set of measurements, which should be collected at each generation, to point out the internal behavior (similarity and difference) between the current optimal-solution to the rest of the population [11][12]. Such information can help us in

understanding of the convergence process within PSO, and guide us on how to better use such algorithms. Also using the distance function we can evaluate and test the effect of increase in the complexity of problem and test the behavior of PSO algorithm [13]. We have classified the distance functions into two groups: an algorithmic DF and a problem-specific DF.

1)*Algorithmic Distance Functions*: The algorithmic distance functions (ADF) are a set of measurements that are directly associated with the evolutionary approaches, such as the convergence rate, executing time, and memory allocation. In this paper, we are more interested in the problem-specific distance function.

2)*Problem-Specific Distance Functions*: The problem-specific distance functions (PSDF) are a set of measurements that are directly associated with the problems to be solved by an evolutionary approach. Therefore, a set of PSDF may not be used to evaluate another problem.

The PSDF defined for our research is X-Diff (read as delta X t), which indicates the root of the sum of the mean square difference between the variable X of the ith individual within the t generation to its best X variable within the same *t* generation, as stated in Equation 3.The term PS represents the population size.

$$\Delta X^{t} = \sqrt{\frac{\sum_{i=1}^{PS} (X_{best}^{t} - X_{i}^{t})^{2}}{PS}}, \forall i = 1,2,3 \cdots PS - 1 \quad (3)$$

Equation 3 states the difference per individual; however, a good PSDF should give an insight behavior of the entire population. The data collected from Equations 6 give us first-order information on how PSO perform at each generation, and how both algorithms are drifting toward the best/optimum [14].

3. Benchmark Functions

A set of benchmark test functions is selected with the goal of representing fitness landscapes ranging from functions in which a global optimum can be found easily using constriction for particle velocity control, to the functions in which it is difficult to find the global optimum. We have used three benchmark optimization functions, which are DeJong F1, Rosenbrock and Rastrigin. Each function has a number of instances depending on the number of unknowns or dimension, *i*. Here we have experimented with five instances for *i* ranging from 2 to 6 per function.

The implementations are scalable, i.e., the dimensions of the functions are adjustable via a single parameter used in the function. The following functions have been used for evaluation.

3.1 De Jong's F1 function (Circle Function)

This function is known as circle function when i = 2 and sphere function in case of i =3. It is continuous, convex and unimodal.

$$Min \quad F(X) = X_{i+1}^{2} + X_{i}^{2}$$

$$Subject \text{ to } \quad -5.12 \leq X_{i} \leq 5.12 \quad (4)$$

Global minimum: $x = 0$; $f = 0$

3.2 Rosenbrock's Valley (Banana function)

Rosenbrock's valley is a classic optimization problem and it is notorious for its slow convergence to the optimum value. The global optimum is inside a long, narrow, parabolic shaped valley. To find the valley is trivial, however convergence to the global optimum is difficult. Hence this problem is often used in assessing the performance of the optimization algorithms. It is a unimodal function and has single local maximum or minimum.

$$Min \quad F(X) = 100(X_{i}^{2} - X_{i+1}^{2}) + (1 - X_{i}^{2})$$

$$Subject \text{ to } -2.048 \leq X_{i} \leq 2.048 \quad (5)$$

3.3 Rastrigin Function

This function is a typical example of nonlinear multimodal function that may have local as well as global optima. It is fairly complex due its complicated search space and large number of local minima.

$$Min \quad F(X) = \sum_{i=1}^{n} (X_{i}^{2} - 10\cos(2\pi X_{i}) + 10)$$

$$Subjet \text{ to } \quad -5.12 \leq X_{i} \leq 5.12 \quad (6)$$

4. Experimental Setting

We have implemented the basic structure of particle swarm optimization algorithm (as depicted in Figure

1) using C language. The DeJong F1, Rosenbrock and Rastrigin functions are used as our examples (as shown in Equations 4, 5, and 6) with 50 individuals in a population and 300 generations for all three functions. The range of each unknown is as shown earlier in equations 4, 5, and 6. The maximum velocity was set always equal to the maximum positive range for each problem. C_1 and C_2 were set to 2. The number of unknowns is increased from 2 up to 6 unknowns for each problem.

5. Results

5.1 PSDF Analysis – DeJong F1 Function

We start with minimum number of unknowns i.e. 2 unknowns for DeJong F1 function which is a circle function. Gradually we increment the dimension to observe the behavior pattern. Figures 2, 3, 4, and 5 display the results obtained for DeJong F1 function for 300 generations, from 2 to 6 dimensions.

As observed from the PSDF values in Figure 2, the individuals' performance is similar for the second unknown in all variations obtained by altering the total number of unknowns. Notice that all individuals start with large diversity but there is huge drop in diversity after say 75 generations and graph is steady plateau.

Considering the second dimension as seen from Figure 3, we plot the values of the second unknown when there are 2, 4, and 6 unknowns respectively. In case of 2 unknowns, the PSDF value of the second unknown is 1.54 in the beginning and after 100 generations it becomes 0.01. When number of unknowns is 4, the value for the second unknown starts with 1.38 and comes to lower than 0.05 within in 100 generations. The individuals behave in similar manner when number of unknowns is 6. The PSDF value for the second unknown was initially 1.02 then it became 0.08 in 100 generations.

When we analyze the graphs for third and fourth dimensions in Figures 4 and 5 respectively, the behavior pattern for PSDF values shows similar curves as it was observed for first and second dimensions respectively. Thus the PSDF values show that the diversity in individuals is high in the beginning of searching then it gradually reduces, which indicates less diversity.

Figure 6 shows the behavior of three unknowns from three different dimensions, namely, 1, 5, and 6. From this figure, we notice that the three unknowns behave in the same way. They all start with unstable and diverse action, and after few generations, all of the unknowns become more stable and less diverse.

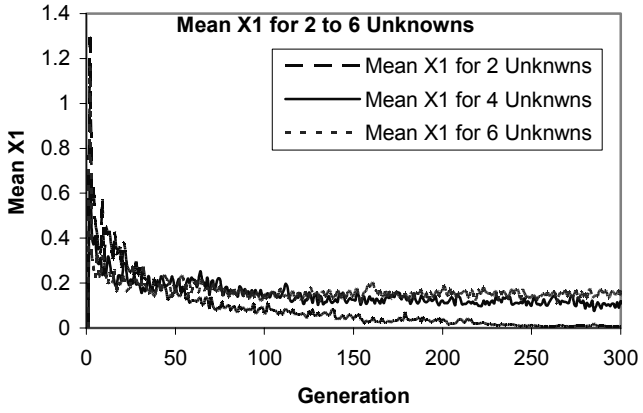

Fig 2 Mean X1 for DeJong F1 Function.

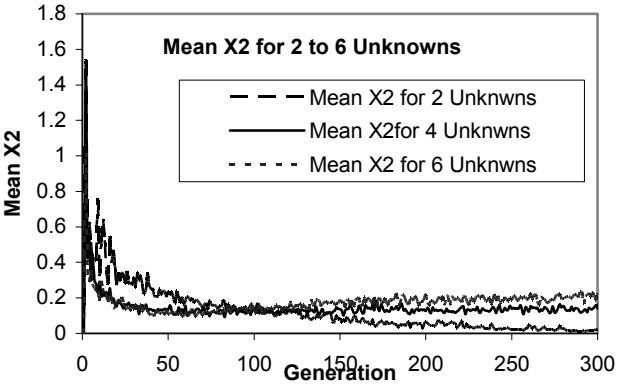

Fig 3 Mean X2 for DeJong F1 Function.

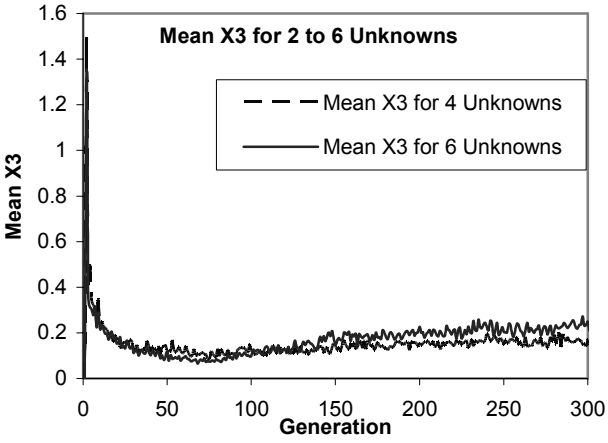

Fig 4 Mean X3 for DeJong F1 Function.

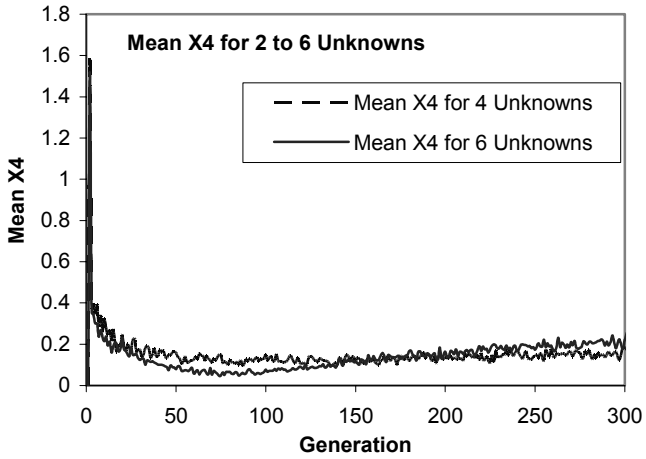

Fig 5 Mean X4 for DeJong F1 Function.

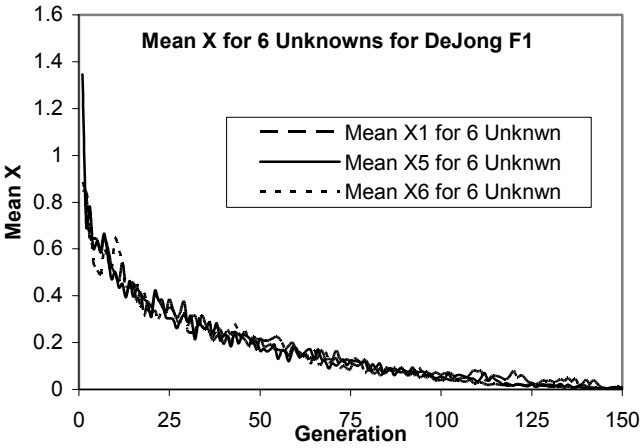

Fig 6 Mean X for DeJong F1 Function in 6th dimension.

5.2 PSDF Analysis – Rosenbrock Function

In the Rosenbrock function, we start with similar setup. We have generated the PSDF values for second dimension and then increasing the complexity of problem, we move to further dimensions. Figures 7, 8, 9, and 11 plots the PSDF values for Rosenbrock function where the numbers of unknowns range from 2 to 6. From these figures, we find that the behavior pattern of individual solutions is in a particular pattern.

From Figure 7, we observed that for first dimension, when number of unknowns is 2, the PSDF value was 1.29 and gradually the diversity between individual solutions reduced and became 0.06 in 100 generations. Also the initial values were respectively 0.67 and 1.06 for 4 and 6 number of unknowns which became 0.14 and 0.13 respectively in 100 generations. Thus the PSO algorithm gains stability in 100 generations for first dimension of Rosenbrock function.

As demonstrated in Figure 8, the values are plotted for second dimension for Rosenbrock function, when number of unknowns is incremented from 2 to 6. In case of 2 unknowns, the value started with 1.54 and came down to 0.11 in just 100 generations. Similarly, initial values for 4 unknowns and 6 unknowns were 1.47 and 1.01 respectively, which decayed to 0.13 and 0.11 respectively, in 100 generations.

As observed from Figures 9 and 10, in case of third and fourth dimensions, it is showing reduction in deviation between individual values and stability after 100 generations. For third dimension the initial values were 1.49 and 1.35 respectively for 4 and 6 unknowns which after 100 generations became 0.11 and 0.12 respectively. However for the fourth dimension, the PSO algorithm generates the values 0.81 for 4 unknowns and 0.87 for 6 unknowns in the beginning. After 100 generations, fourth dimension values become 0.29 and 0.37 for 4 and 6 number of unknowns, thus gaining stability in fewer generations.

Thus, the deviation in individual positions is more initially, but more or less PSDF has same value for number of unknowns equal to 3, 4, 5 or 6, and this deviation drops to approximately 0 which is optimum value in less than just 150 generations. Thus irrespective of the number of unknowns for the given problem the plot is same.

In figure 11, observation of the behavior pattern of PSO algorithm when applied to Rosenbrock is shown. We plot the graph for 5^{th} and 6^{th} dimensions, and analyze it. From the analysis, it is observed that the individuals gradually converge with reduction in diversity in both dimensions.

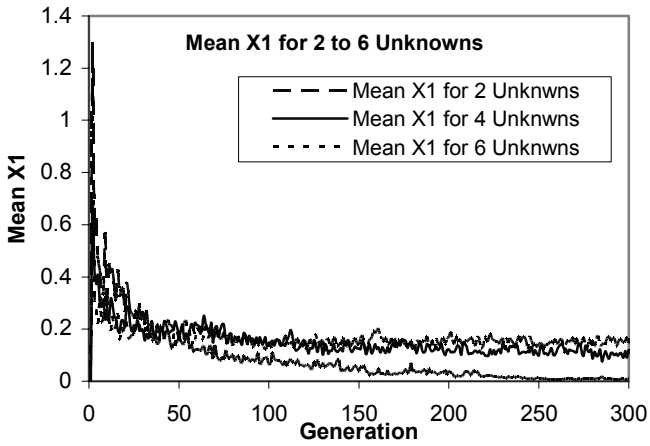

Fig 7 Mean X1 for Rosenbrock Function.

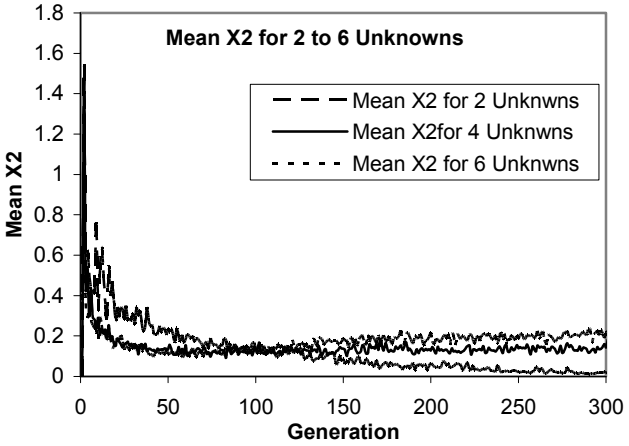

Fig 8 Mean X2 for Rosenbrock Function.

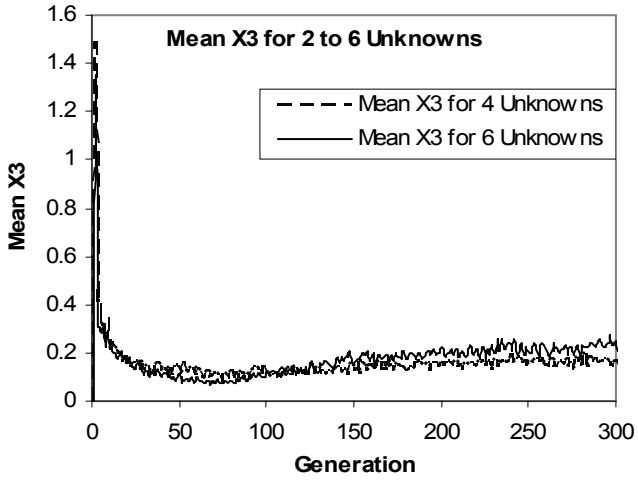

Fig 9 Mean X3 for Rosenbrock Function.

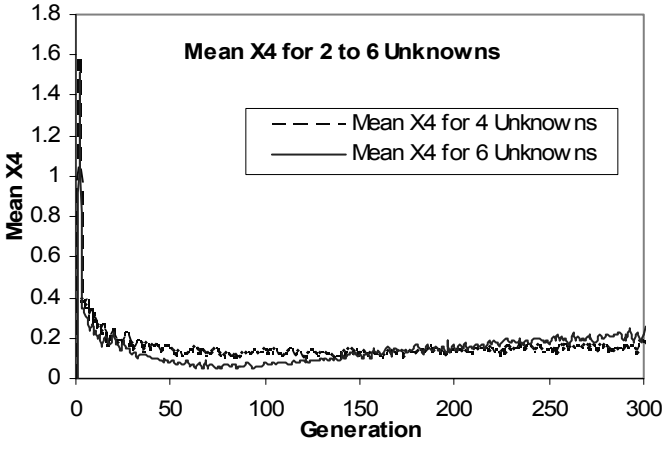

Fig 10 Mean X4 for Rosenbrock Function.

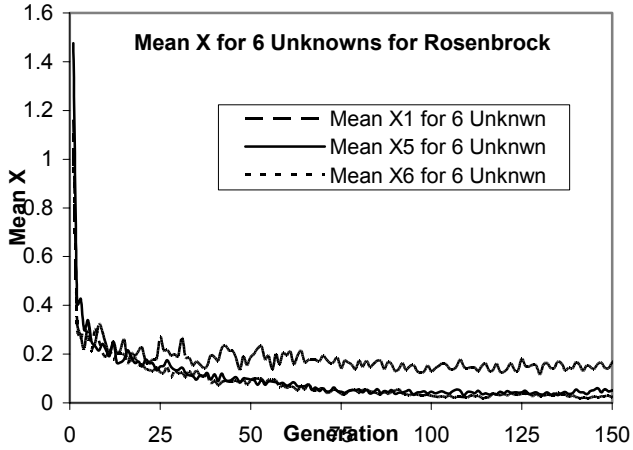

Fig 11 Mean X for Rosenbrock Function in 6th dimension.

5.3PSDF Analysis – Rastrigin Function

We started with first dimension and consequently analyzed second, third and fourth dimensions, in case of Rastrigin function.

Figures 12, 13, 14 and 15 are plot of PSDF values for parameters ranging from 2 to 6 unknowns for the benchmark function of Rastrigin. As observed, though the initial value is large for all the three cases, the optimum value is almost reached in less than 100 generations, irrespective of variation in number of unknowns. The PSO algorithm gains stability soon in approximately 100 generations as was the similar case with DeJong F1 function and Rosenbrock function.

As observed from Figure 12, within only 100 generations, the values converge, even though they started at values as high as 1.35, 2.47 and 1.01 for 2, 4, and 6 unknowns respectively. The diversity in individual vales reduces drastically and come down to 0.16, 0.26 and 0.31 in 100 generations.

Figure 13 shows the second dimension values for 2, 4, and 6 unknowns. The graph starts with values 1.12, 2.47, and 0.99 for 2, 4, and 6 unknowns respectively. After 100 generations, the same values for 2, 4, and 6 generations are 0.13, 0.30, and 0.32 respectively.

From Figure 14, we get the values for third dimension, which are 1.31 and 0.87 respectively for 4 and 6 unknowns initially. The values decay to 0.26 and 0.36 for 4 and unknowns respectively in 100 generations.

As observed from Figure 15, after 100 generations, the values for fourth dimension for 4 unknowns is 0.29 which was initially 0.81, and for 6 unknowns it is 0.37 which was initially 0.86. Thus

the graph is steady plateau and individuals are not deviated at all after few generations.

Figure 16 also justifies our observation that when values are graphed for Rastrigin function, with as much complexity as in 6th dimension, the values for first, fifth and sixth dimensions are plotted and it is showing a constant decay and finally forms a plateau in less than 100 generations where diversity is less.

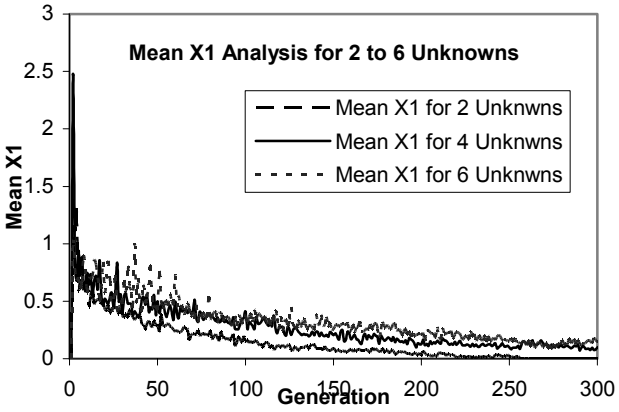

Fig 12 Mean X1 for Rastrigin Function.

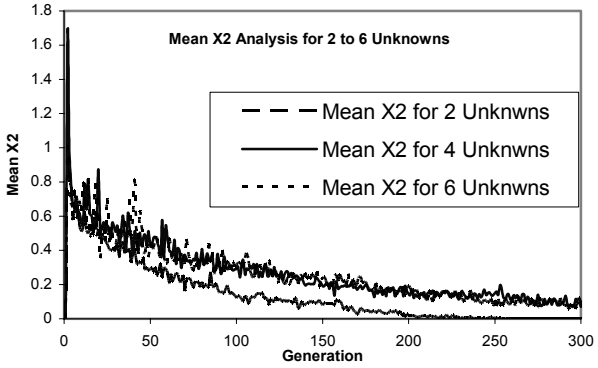

Fig 13 Mean X2 for Rastrigin Function.

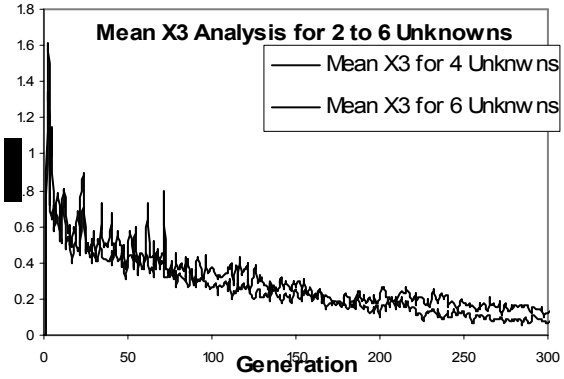

Fig 14 Mean X3 for Rastrigin Function

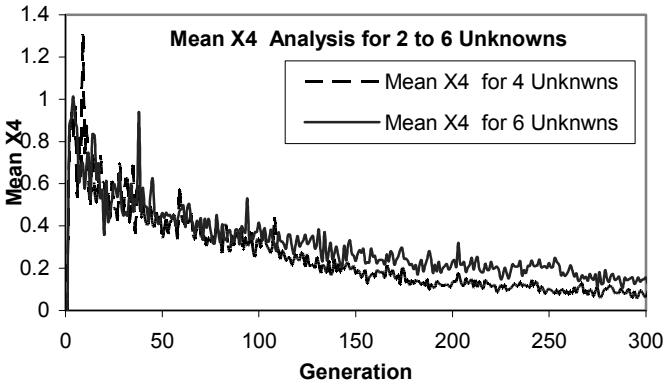

Fig 15 Mean X4 for Rastrigin Function.

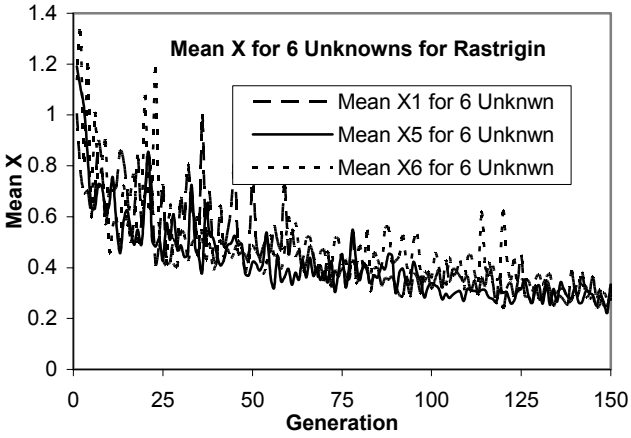

Fig 16 Mean X for Rastrigin Function in 6th dimension.

6. conclusion

This work analyzes internal behavior of the PSO algorithm, with respective increment in the number of unknowns which means the complexity of problem-domain, using the values of Problem Specific Distance Function (PSDF) for the three well-known mathematical benchmark functions, such as DeJong F1, Rastrigin and Rosenbrock. The criterion for analysis was variation in the number of unknowns for these benchmark functions from 2 unknowns to 6 unknowns. We observed that individuals behave in similar manner irrespective of the complexity of the problem. The mean distance function started with a large value, but it converged to optimum in few generations. That means, once we know the results for 2 or more number of unknowns, we can predict the expected behavior for incremental number of unknowns. We have justified the behavior using all the three objective benchmark problems under study.

The parameters V_{max} and the range of variables of PSO are fixed during the entire experiment. Thus increasing the complexity of a problem, and with the fixed settings for PSO parameters, the algorithm behaves in similar manner which means that PSO algorithm is stable. In addition, the result was that the algorithm always start with big collection of individuals and bring all the individuals together near the optimum value, thus becoming less diverse. This was true for all the dimensions, which means that no matter how complex the problem becomes, PSO behaves the same. As a conclusion, if we have very complex problem with many unknowns, testing it for few unknowns will give us enough prediction whether PSO is able to solve this problem or not. Thus we can use PSDF as a measure whether PSO can solve the problem in hand or not. This will be done by applying the PSDF analysis to less complex problem and observe the behavior of PSO to it. If PSO was able to converge to the optimum, then we can directly use it for more complex cases.

From this experiment, we can observe that PSO has a convergence rate which is quite high, the stability is achieved in fewer generations, and the diversity of individuals reduces significantly. In addition, PSDF values helped in predicting whether PSO will be able to solve more complex cases or not. In future work, we will try to find the actual values of unknowns in more complex dimension using PSO, if we have the values of unknowns in less complex problem.

References

[1] P. J. Angeline, "Evolutionary Optimization versus Particle Swarm Optimization: Philosophical and Performance Differences," in *Proc. Evolutionary Programming VII (EP98,)* LNCS 1447, pp. 601-610, 1998.

[2] A. Czarn, C. MacNish, K. Vijayan, B. Turlach, and R. Gupta, "Statistical Exploratory Analysis of Genetic Algorithms," IEEE *Transactions on Evolutionary Computation*, vol. 8(4) , August 2004, pp. 405-421, 2004

[3] B. Cestnik, "Estimating probabilities:A Crucial Task in Machine Learning," in *Proc. T*he *European Conference in Artificial Intelligence*, pp.147-149.

[4] Y. Shi and R. Eberhart, "Parameter Selection in Particle Swarm Optimization," *Proc. EP98*, 1998.

[5] B. Jung and B. Karney, "Benchmark Tests of Evolutionary Computational Algorithms," *Environmental Informatics Achieves*, vol. 2, pp. 731-742, 2004.

[6] W. Chang, A. Sutcliffe and R. Neville, "A Distance Function-Based Multi-Objective Evolutionary Algorithm," in James Foster (editors), Genetic and Evolutionary Computation Conference. Late-Breaking Papers, AAAI, Chicago, Illinois, USA, pp. 47--53, 2003.

[7] S. Louis, "Genetic Learning for Combinational Logic Design," *Journal of Soft Computing manuscript*, vol. 9(1), pp. 38–43, 2004.

[8] J. Holland, "Adaptation in Natural and Artificial Systems," The University of Michigan Press, 1975.

[9] X. Hu, "PSO Tutorial," http://www.swarmintelligence.org/tutorials.php.

[10] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proc. the IEEE International Conference on Neural Networks*, 1995.

[11] S. Manay and A. Yezzi, "A Second-Order PDE Technique to Construct Distance Functions with More Accurate Derivatives," in *Proc.* IEEE International Conference on Image Processing, *2003*, vol. 1, pp. I - 873- 6, 2003.

[12] X. Yao, "How Powerful/Efficient Is Your Evolutionary Algorithm," A keynote Presentation at 2004 IEEE Congress on Evolutionary Computations, Portland, Oregon, USA, 2004.

[13] Y. Shi and R. Eberhart, "A Modified Particle Swarm Optimizer," in *Proc. the IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, pp. 69-73, 1998.

[14] S. Habib and B. Al-kazemi, "Comparative Study Between the Internal Behavior of GA and PSO Through Problem-Specific Distance Functions," in *Proc. 2005 IEEE Congress on Evolutionary Computation*, 2005.

[15] K. McAloon and C. Tretkoff, "Optimization and Computational Logic," *Wiley Interscience Series in Mathematics and Optimization.* Wiley Press, New York, 1996.