International Journal of Hybrid Information Technology
Vol. 9, No. 5 (2016), pp. 165-172
http://dx.doi.org/10.14257/ijhit.2016.9.5.13

Analysis of Performance of Particle Swarm Optimization with Varied Inertia Weight Values for solving Travelling Salesman Problem

Ashima Chopra[1] and Mandeep Kaur[2]

[1]Computer Science
[1]Guru Nanak Dev University,
RC Jalandhar, Punjab, India
[2]Computer Science,
[2]Guru Nanak Dev University
RC Jalandhar, Punjab, India

Abstract

Particle Swarm Optimization is a popular heuristic search technique developed by Eberhart and Kennedy in 1995 which takes its inspiration from the social and cognitive learning of birds or fishes. This algorithm comprises the involvement of swarm intelligence technique for optimization. The most widely accepted variation of the basic PSO technique is PSO with Inertia weight which substantially controls the convergence behaviour and exploration exploitation trade-off in the basic PSO technique. From its initialization onwards a huge range of modifications of Inertia Weight strategy have been recommended. This paper involves the use of PSO with varying values of inertia weight for solving the Travelling Salesman Problem. An analysis of how different inertia weight values effect the solution in terms of time complexity, space complexity and convergence in carried out in order to know the value best suited for setting up the inertia weight.

***Keywords**: Particle Swarm Optimization, Inertia weight, Run-time, Convergence, Space Complexity*

1. Introduction

Particle swarm optimization technique was first brought into practice by Kennedy and Eberhart in 1995 [1]. This technique is an analogy to the behaviour of swarm of birds or a group of living organisms moving here and there in the search of food. Based on the movement laws followed by the swarm this technique aims at providing the global best solution. The particles of the swarm referred to as points in search space move multidimensionally in hunt for an optimal solution .The movement and behaviour of each particle is controlled by its own experience and through interactions with the neighbour particles in the swarm. Due to its simple definition and promising results this optimization method attracted a huge range of researchers from varied fields.

The basic PSO [1] for some problems may suffer from premature convergence or may require greater runtime to generate results. Thus, basic PSO due to its shortcoming paved ways for the researchers to introduce its improved version. Various different variants of the basic technique were developed with the major task of improving the performance of PSO by maintaining the balance between exploration-exploitation [5]. The inertia weight, introduced in 1998 [3], serves as a key component in achieving the above mentioned goal of balancing exploration and exploitation.

2. Problem Formulation

PSO is heuristic technique employed to find a best solution much faster than the other algorithms. It's simple concept and easy implementation has made it popular among researchers. It suffers from the problem of swarm stagnation. Swarm stagnation occurs when the best solution is located at local optima trapping the whole swarm leading to swarm stagnation. . Thus, to overcome this short coming of PSO its varied version PSO with inertia weight was introduced. This paper analyses how the varying value of inertia weight affects the solution of TSP problem which is solved using PSO with inertia weight technique. The analysis focuses on finding out the value of inertia weight that is optimal from the point of view of convergence, execution time and space complexity.

The travelling salesman problem (TSP) is included in the group of most widely evaluated NP-hard combinatorial problem .Its definition is easily understandable to a naive user because of its deceptively simple definition .Yet it remains in the elite group of one of the most challenging problems in research. The simple description of TSP is:

Given a list of cities and their intermediate distances, the aim is to find a shortest possible path that covers each city exactly once. Assume a graph G = [v, e] where v represents set of vertices and e represents set of edges. Consider D_{ij}= d as the distance matrix associated with each edge e. The TSP provides solution in the form of a minimum cost or distance cycle known as Hamiltonian circuit or cycle that traverses each vertex exactly once.

3. Particle Swarm Optimization

PSO is a population based optimization technique which comprises of number of mass less and volume less particles that can be called as points that flies in the search space towards the best solution in the swarm. The movement of the particles in the swarm space is governed by the socio-psychological model of learning. PSO guides the swarm towards optimal solution by allowing information exchange among the individual particle of the swarm. Two vectors *i.e.*, velocity vector and a position vector are being maintained by each particle of the swarm. It is mandatory for each of the particle to follow velocity and position update rules during each generation. The updations are made possible by knowing the particle's previous best position and the best position found by the entire swarm so far. Let vi and xi be the velocity and the position vector respectively and M be the no of particle in the search space or swarm. The update rules in the standard PSO [1] are defined as

$$v_i^j \leftarrow v_i^j + c_1.r_1^j.(pbest_1^j - x_1^j) + c_2.r_2^j.(gbest_1^j - x_1^j) \quad (1)$$

$$x_1^j \leftarrow x_1^j + v_1^j \quad (2)$$

In equation. 1, pbest is the best position of a particle whereas gbest is the best position of the whole swarm. c_1 and c_2 are the two constants to measure relative performance of pbest and gbest. r_1^j and r_2^j are random numbers distributed in [0, 1] and j (1<j<n) represents jth dimension of the search space. In PSO, if the particles get confined to local minimum, there exists a tendency for escape via a sort of momentum built into the algorithm via inertia weight parameter.

4. Inertia Weight Particle Swarm Optimization

To date, many variants of the basic PSO have been proposed to improve its PSO. In order to bring about balance in the global search and local search capabilities of PSO, Shi

et al. introduced a new parameter called inertia weight into basic PSO [6]. This algorithm is called inertia weight PSO (IWPSO) and shows improvement in terms convergence speed in comparison with the original PSO.
In IWPSO, the velocity update equation (1) of basic PSO is modified as follows:

$$v_i^j \leftarrow wv_i^j + c_1.r_1^j.\left(pbest_1^j - x_1^j\right) + c_2.r_2^j.\left(gbest_1^j - x_1^j\right) \qquad (3)$$

where, w represents the inertia weight. In IWPSO, the function of inertia weight w is to balance global exploration and local exploitation. It is also meant for controlling the previous velocity's impact on the present velocity for a given particle. Inertia weight w linearly decreases according to Equation 4.

$$w = w_{max} - \frac{w_{max} - w_{min}}{k_{max}} \times k \qquad (4)$$

where w_{max} is the initial weight, w_{min} is the final weight and k_{max} is the maximal iteration numbers and k is the current iteration number.

Need for Inertia weight in Particle Swarm Optimization:

- PSO suffers from swarm stagnation where
- in the particles of the swarm get confined to local minimum. With the advent of inertia weight there exists a scope for the particles to escape with the onset of momentum built into the algorithm via inertia weight parameter.

- The motivation behind the inception of inertia weight was to eliminate the need for a constant v_{max} which was introduced in the basic PSO mainly to limit the velocities of the particles and improve the resolution of the search and to act as a constraint for controlling the global exploration ability of particle swarm. Further, the concept of an Inertia Weight was developed to better control exploration and exploitation and to eradicate the need forV_{max}.

- As initial velocity plays a pivotal role in balancing exploration and exploitation process of swarm Inertia Weight (w) is used to control the velocity.

- In IWPSO the inertia weight controls the impact of the previous velocity on the present velocity for a particle which in turn is helpful for maintaining the balance among global exploration and local exploitation.

5. Experimental Parameter Setting

- Initial Population

The initial populations are generated at random and initialized to n particles at time t.

- Swarm

It is an apparently unregulated pattern of moving particles wherein each particle is visualized as moving in a haphazard manner in random direction.

- Population Size

The performance of the standard algorithm is independent of the population size as concluded from the past research performed by Eberhart and Shi [18]. Therefore, in order to reduce the computational requirements the population size in this analysis is fixed to 10 particles.

- Maximum Iterations

This refers to the maximum number of iterations carried out in order to obtain the optimal result by allowing the convergence of the fitness value with the optimal result possible. In this analysis, the maximum iterations are set as 200.

- Inertia Weight Considerations

The PSO algorithm used in this paper uses the value for the inertia weight initiating from 0.45, 0.55, 0.65, 0.75, 0.85, and 0.95 for solving the TSP.

6. Experimental Results

In order to describe the effect of change in inertia weight values on the efficiency of the optimization results the algorithm was tested using several values of inertia weight [0.45, 0.55, 0.65, 0.75, 0.85, and 0.95] for solving the TSP problem. In this study, we analysed the convergence, execution time and space requirements obtained using the different inertia weight values. Based on the results obtained the optimal value for inertia weight in terms of run-time, memory requirements and convergence are demonstrated.

The experimental were performed on Intel(R) Core(TM) i5-2430M CPU 2.40GHz/4G RAM Laptop using MATLAB R2012a.

a. Analysis based on Run-time

The graphical figure presented provides the completion time of the TSP corresponding to the particular inertia weight value. From the experimental results obtained it can be clearly concluded that with the increasing value of inertia weight the completion time of TSP increases. From the experimental results it can be clearly intercepted that for inertia weight value 0.45 the algorithm competes in 23.1057 seconds which is the minimum value and the maximum value i.e. 25.7739 seconds is taken by the inertia weight value 0.95.

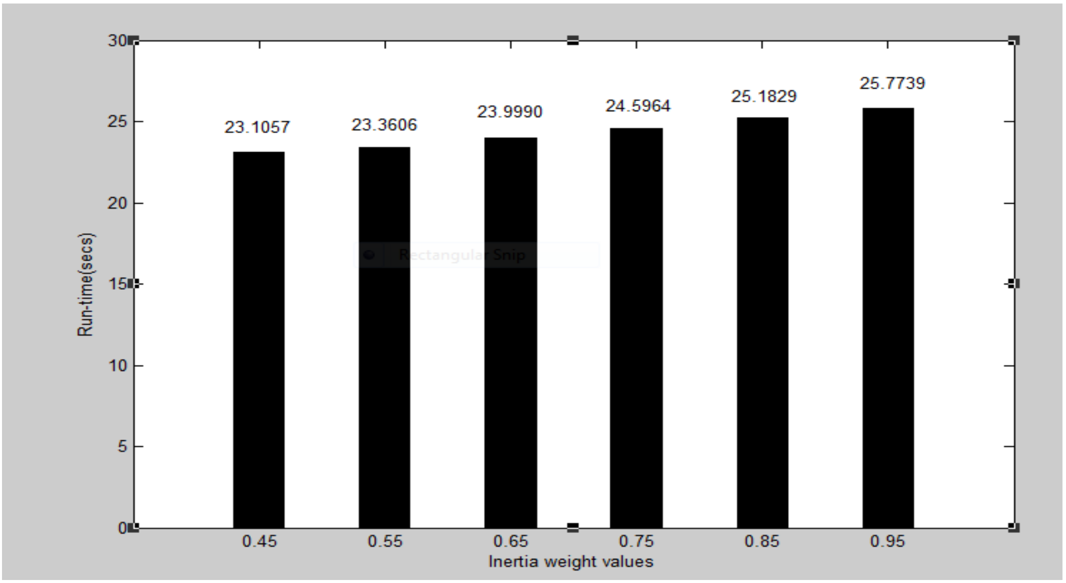


Figure 1. Graphical Representation of affect of Varied Inertia Weight Value on Run-Time

b. Analysis based on the space requirements

The table below enlist the memory requirements for the algorithm for each different value of inertia weight. The space needed in most cases is equivalent and on an average is in the range of 0.0690 bytes to 0.0700 bytes. On comparative grounds, it can be analyzed that the least memory requirements are for the inertia weight value 0.55 and inertia weight 0.65 occupies the most of the space.

Table 1. Analysis of Memory Requirements for Varied Inertia Weight Values

Inertia weight values	Memory space occupied (bytes)
0.45	0.0694
0.55	0.0690
0.65	0.0700
0.75	0.0698
0.85	0.0690
0.95	0.0695

c. Analysis based on convergence speed

The following figures show the convergence comparisons. The experimental results show that the convergence speeds of PSO algorithm shows the best performance for inertia weight value as 0.55 and 0.85 and the worst performance is observed for inertia weight value 0.45.

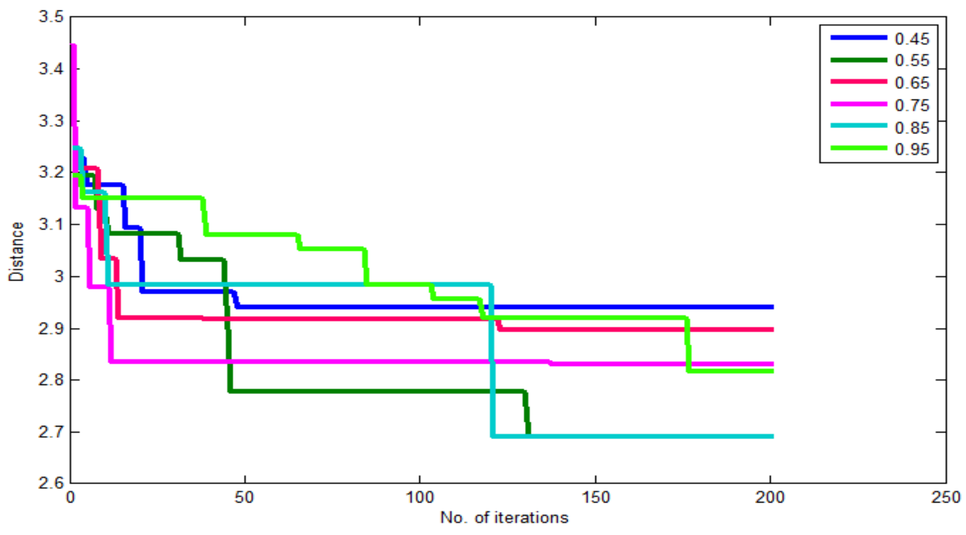
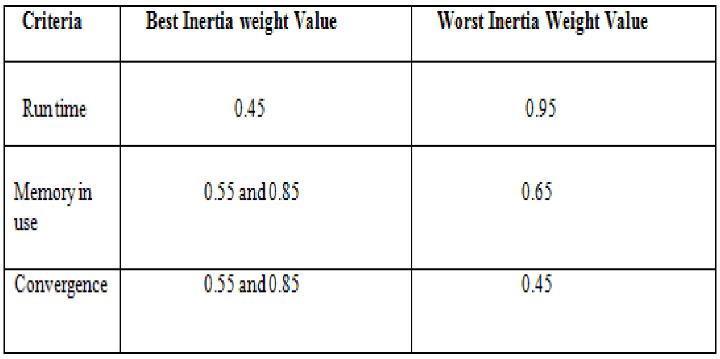


Figure 2. Analysis of Convergence Pattern for Varied Inertia Weight Value

7. Conclusion

This paper presents an analysis of the scenario in which the varied values of inertia weight are taken into consideration for checking the performance of PSO. To make the results more specific in nature inertia weight PSO algorithm is used for solving the classical TSP problem. A precise evaluation of performance of PSO is carried out on the basis of run-time, memory requirements and convergence trends obtained for different inertia weight values. From the experimental results it can be undoubtedly observed that on increasing the inertia weight value the run-time increases. For space complexity it can be analysed that the space requirement for all inertia weight values lie within the range of 0.0690 and 0.0700 with the best case result for inertia weight value 0.55 and worst case results. In case of convergence speed the algorithm shows best convergence when inertia weight value is taken as 0.55 and 0.85 and shows the worst convergence when the inertia weight value is taken to be 0.45.

Table 2. Conclusions Derived from the Experimental Results Obtained



Criteria	Best Inertia weight Value	Worst Inertia Weight Value
Run time	0.45	0.95
Memory in use	0.55 and 0.85	0.65
Convergence	0.55 and 0.85	0.45

Acknowledgment

History of all great works is a clear indication that no great work was ever done without either the active or passive support of a person's surroundings and one's close quarters. This research paper is made possible through the help and support from everyone, including: parents, teachers, family, friends, and in essence, all sentient beings. Especially, I would like to dedicate my acknowledgment of gratitude toward the following significant advisors and contributors: First and foremost, I would like to thank Er. Mandeep Kaur for her support and encouragement. She kindly read my paper and offered invaluable detailed advices on organization and the theme of the paper. Finally, I sincerely thank to my parents, family, and friends for their consistent the advice and support. The product of this research paper would not be possible without all of them.

References

[1] R.C. Eberhart and J. Kennedy, "Particle Swarm Optimization", IEEE Conference on Neural Networks, **(1995)**.
[2] Y. Shi and R.C. Eberhart, "Particle Selection in Particle Swarm Optimization", **(1998)**.
[3] R.C. Eberhart and Y. Shi, "Modified Particle Swarm Optimizer", IEEE Conference on Evolutionary Computation, **(1998)**.
[4] P.N. Suganthan, "Particle Swarm Optimizer with Neighbourhood Operator", Congress on Evolutionary Computation, **(1999)**.
[5] Y. Shi and R.C. Eberhart, "Empirical Study of Particle Swarm Optimization", Congress on Evolutionary Computation, (**1999**).
[6] J. Kennedy and M.Clerc, "Particle Swarm Explosion, Stability, Convergence in Multi-Dimensional Space", **(2002)**.
[7] R. Mendes and J. Kennedy, "The Population Structure and Particle Swarm Performance", Congress on Evolutionary Computation, **(2002)**.

[8] Y. Shi and R.C. Eberhart, "Tracking and optimizing Dynamic Systems with Particle Swarm", **(2001)**.
[9] F.V. Bergh and A.P. Engelbrecht, "A Cooperative Approach to Particle Swarm Optimization", **(2004)**.
[10] M.S. Arumugam and M.V.C. Rao, "The Performance of Particle Swarm Algorithm with Inertia Weight Variants for Computing Optimal Class of Hybrid Systems", **(2006)**.
[11] M. Iwamatsu, "Locating All the Global Minima using Multi-species Particle Swarm Optimizer: The Inertia Weight and the Constriction Factor Variants", IEEE Congress on Evolutionary Computation, (**2006)**.
[12] Y. Feng, X, A.X. Wang, G.F. Teng and Y.M. Yao, "Chaotic Inertia Weight Particle Swarm Optimization", Innovative Computing Information and Control, **(2007)**.
[13] R.F. Malik, T.A. Rahman. R. Nhah and S.Z.M. Hashim, "New Particle Swarm Optimization with Sigmoidal Increasing Inertia Weight", International Journal of Computer Science and Security, **(2007)**.
[14] M.O. Lin, Zheng Hua, "Improved PSO Algorithm with Adaptive Inertia Weight and Mutation", World Congress on Computer Science and Information Engineering, (**2009)**.
[15] J. Xin, G. Chen and Y. Hai, "The Particle Swarm Optimization with Multi-Stage Linear Decreasing Inertia Weight", Computational Sciences and Optimization, **(2009)**.
[16] Sameh Kessentini and Dominique Barchiesi, "Particle Swarm Optimization and Adaptive Inertia Weight", International Journal of Machine Learning and Computing, (**2015)**.
[17] K. Kentzoglanakis and M. Poole, "Particle Swarm Optimization with Oscillating Inertia Weight", Conference on Genetic and Evolutionary Computation, **(2009)**.
[18] R.C. Eberhart and Y. Shi, "Comparison between Genetic Algorithms and Particle Swarm Optimization", Evolutionary Programming", **(1998)**.