Test Effort Estimation-Particle Swarm Optimization Based Approach.

CONFERENCE PAPER · JANUARY 2011

DOI: 10.1007/978-3-642-22606-9_46 · Source: DBLP

4 AUTHORS, INCLUDING:

Aloka Sudhodanan
IBM
3 PUBLICATIONS **6** CITATIONS

SEE PROFILE

Dr. Praveen Ranjan Srivastava
Indian Institute of Management, Rohtak
84 PUBLICATIONS **339** CITATIONS

SEE PROFILE

Test Effort Estimation-Particle Swarm Optimization Based Approach

S. Aloka, Peenu Singh, Geetanjali Rakshit, and Praveen Ranjan Srivastava

Computer Science & Information System Department, BITS PILANI – 333031 India
{alokas88,geetanjali.rakshi,singhpeenu,
praveenrsrivastava}@gmail.com

Abstract. Test Effort Estimation is an important activity in software development because on the basis of effort cost and time required for testing can be calculated. Various models are available for estimating effort but to some extent all models result in erroneous effort estimation. So there is a need to optimize the effort estimated. Meta heuristic techniques can be used for this purpose, to optimize a problem by iteratively trying to improve a solution, using some computational methods. Particle Swarm Optimization is one such technique which have been incorporated in this work to get good test effort estimates.

Keywords: Particle Swarm Optimization (PSO), Test Point Analysis (TPA), Use Case Points (UCP).

1 Introduction

Software engineering is a systematic approach for developing high-quality software in a cost-effective manner [1]. In software development process, software testing is an important and complex issue. According to literature survey, software testing consumes the maximum effort, which is nearly 50% of the total effort. Test Effort Estimation is an important activity in software development because estimating the effort beforehand enables project managers to allocate resources i.e. budget, time and staff efficiently and avoid future inconvenience [1]. Currently, many effort estimation techniques are available giving satisfactory estimates. However, with increasingly tight schedules and market competition, more accurate estimates are needed.

To handle the complex nature of software engineering, various meta-heuristic techniques[3] like Tabu Search[4] and soft computing techniques[5], are used. Particle Swarm Intelligence is one of the meta-heuristic techniques which is widely used for optimization in various fields. In this paper, results of the two existing techniques, use case points (UCP) [6] and test point analysis (TPA) [7] are optimized to achieve greater precision in test effort estimation using Particle Swarm Optimization (PSO).

The following sections discuss the related works in this field, proposed strategy to apply PSO to optimize the estimates, the results obtained on applying this on a case study, one each for Use Case Points and Test Point Analysis. Then these results are compared with those obtained from existing methods. Finally future scope for this work is discussed.

S. Aluru et al. (Eds.): IC3 2011, CCIS 168, pp. 463–474, 2011.

2 Background

Many development effort estimation techniques, like COCOMO model [1], are available from which test effort can be estimated to be nearly 50% of development effort. However, for estimation of test effort directly, not many approaches are available at present. Few techniques which are used are test point analysis and use case points. Suresh Nageswaran proposed use case point analysis technique for test effort estimation [6]. In this technique, test effort is estimated on the basis of use cases and actors of the system by rating them into different categories. Some technical factors are also considered.

Another method is Test point analysis by Drs Eric P W M Van Veenendaal CISA and Ton Dekkers [7], where software is divided into different modules and for each module, some parameters are considered to calculate total test hours.

Results obtained from these techniques can be further optimized to give more accurate results. Various evolutionary techniques have been successfully applied earlier, like software effort estimation using neural networks [8], software effort estimation using soft computing techniques [5] and using Fuzzy Logic [9].

However, Particle Swarm Optimization is known to have some advantages over these techniques. It is much easier to understand and implement due to less number of parameters and has,less sensitivity to the nature of objective function, is less dependent on initial points in search space and quickly converges giving stable and high quality results. Its CPU and memory requirements are also less.

So, PSO algorithm has been used to optimize the effort obtained by Use Case Point Analysis and Test Point Analysis. Unlike other estimation techniques, these consider various parameters related to software testing, giving good results. The results need to be more refined, so optimization using PSO is done.

3 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is an optimization technique developed by Dr. Eberhart and Dr. Kennedy [14], based on social behavior of bird flocking or fish schooling [10]. PSO has many similarities with evolutionary computation techniques like Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation [11]. In PSO, the potential solutions in the search space form particles in the swarm. Each particle has a position and a non-zero velocity at any instant, and is aware of the global best and the local best(self best) positions. They fly through the problem space by following the current global and local best, and approach towards optimality.

4 Use Case Points (UCP)

In Use Case Points following approach is used for effort estimation [6, 12]:

1. First of all number of actors in the system are determined and categorized into 3 levels : simple, average and complex, based on their complexities. Then weights are

assigned to each type of these actors. From this Unadjusted Actor Weight (UAW) is calculated.
2. A similar procedure is applied to use cases in the system and Unadjusted Use Case Weights (UUCW) is determined.
3. Unadjusted Use Case Points is calculated using the formula UUCP=UUCW+UAW
4. Technical and Environmental Factors (TEF) are computed.
5. Adjusted UCP is calculated as: AUCP = UUCP*[0.65+0.01*TEF]
6. Final effort is computed as: Final effort = AUCP* ratio of development man-hours needed per use case point.

5 Test Point Analysis (TPA)

Test point analysis is a technique to measure the black box test effort estimation. TPA calculates test effort estimation for the functions in the system, and also for the whole system, in terms of test points. These test points are calculated based on the importance assigned to the functions by the user [7, 13]. Quality characteristics like functionality, usability, security, efficiency, etc are taken into account in this technique with proper weights assigned to each.

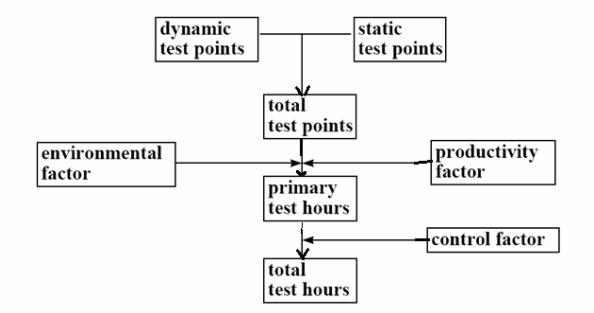

Fig. 1. TPA Technique

As shown in Fig.1. first Dynamic Test Points and Static Test Points are calculated. Next, total test points are obtained by adding dynamic and static test points. Then Primary Test Hours are calculated using Total Test Points, Productivity Factor and Environmental Factor. Finally, Total Test Hours are calculated using primary test hours and planning and control allowance.

Though both UCP and TPA give good results, there is still a lot of gap between the actual and predicted values. The results from these methods can be further refined. This is an attempt to optimize the results of these two techniques using PSO.

6 Proposed Strategy

This work has been carried out to optimize test effort estimation in order to reduce the difference between actual and predicted effort. To achieve this aim, Particle Swarm

Optimization is applied on both the methods, TPA and Use Case, in the following manner as explained in Fig.2.

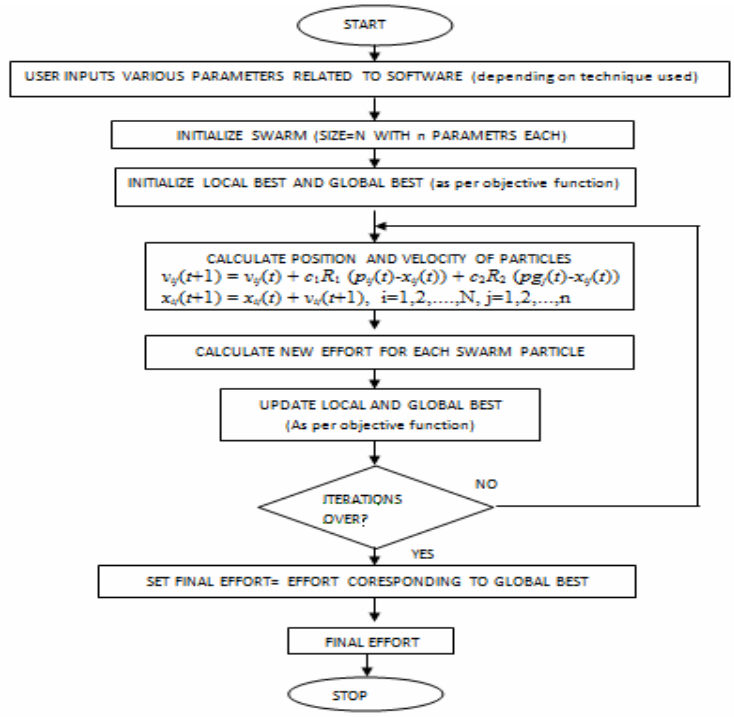

Fig. 2. Solution Model

The solution model flowchart explains the strategy followed in the optimization process. The steps included are initialization of swarm, application of PSO algorithm and analysis of the obtained output. The steps are explained below in detail:

1. A search space of 10 particles (where particles represent possible solutions in the search space) is considered. The study was conducted in different swarm sizes, and with a swarm size of 10 particles it was found through simulation that more optimal solutions were derived. Hence a search space of 10 particles was taken into consideration. Here particles are total effort calculated by randomly initialized weights for different ratings in that particular method (UCP or TPA).
2. Each particle is represented as a function of various parameters of the respective technique used, like UCP and TPA.

In use case 16 parameters are present (3 for actor weights, 4 for use cases and 9 for technical factors) [6] In Test Point Analysis 24 parameters related to Function Dependency, Environmental Factors, Static and Dynamic Quality characteristics, Team size and Planning and Control tools are taken into account [7].

3. These particles will move in search space to achieve global best according to PSO equations [11]:

Velocity of the particle is updated by the equation:

$$v_{ij}(t+1) = \omega v_{ij}(t) + c_1 R_1 (p_{ij}(t)-x_{ij}(t)) + c_2 R_2 (pg_j(t)-x_{ij}(t))$$

Position of the particle is updated by the equation:

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1).$$

where v_{ij} and x_{ij} are vectors representing the velocity and position of the particle respectively. Here, with respect to this problem, position is the weights/values assigned to the parameters and velocity is the rate of change of these values. Suppose, n is weight assigned to simple actor in use case analysis. Then, n is the present x_{ij}. Then after iteration, if v_{ij}=2, the weight becomes n+2, that is, new x_{ij}.

c_1, c_2 are constants representing cognitive and social parameters, respectively. The combination of these two parameters determines the property of convergence of the algorithm. The portion of the adjustment to the velocity influenced by the local best is considered to be the cognitive component, and the portion influenced by the global is the Social component.

ω is the inertia factor, introduced to give best PSO performance by linearly decreasing its value as it moves towards optimized results[15].

R_1, R_2 are random numbers between 0 and 1. $p_{ij}(t)$ is the local best, which is the best value attained by an individual particle till the time t. $pg_j(t)$ is the global best, which is the best of all values obtained in the entire swarm till the time t. In each iteration, the position and the velocity of the particle as well as the local best and global best are updated according to the above equations.

According to these equations, the estimated effort value will move in the search space trying to reach the optimal value in each iteration. These equations are applied iteratively until particles converge to optimal solution, that is, where effort is closer to actual value. In every iteration, each particle has its local best (best position achieved by that particle in its lifetime). Best position from all the local best is the global best and movement of particles is dependent on both local and global best.

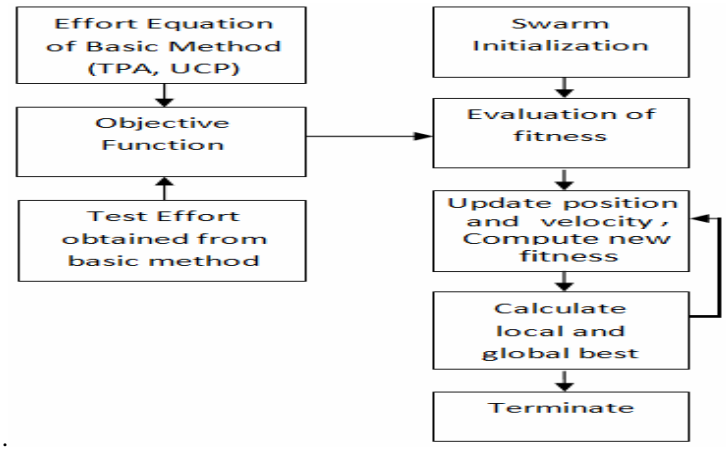

Fig. 3. Architecture

4. The objective is to get optimum effort closer to the one obtained by applying basic method. The objective function is selected based on what is to be optimized. Here since optimization of test effort estimation is considered, the basic equations of estimating test effort (in UCP and TPA) are taken as the objective function. So, objective function uses basic equation for effort estimation in that particular technique (TPA or Use Case). This objective function determines local and global best.

5. As an output, optimized effort is obtained.

The overall architecture of this algorithm is shown in Fig 3, which shows that optimal results can be obtained after applying PSO on swarm. It shows the objective function derived from TPA and UCP equation and effort obtained from these methods is used to evaluate fitness of the swarm particles. Then as PSO works iteratively and results converge to an optimal solution.

7 Case Study

Example 1. Use Case Points Analysis

The following example is taken from the paper "Test Effort Estimation Using Use Case Points" by Suresh Nageswaran [6]. The example considered here is that of a web-based software. The use cases as well as actors were identified from the requirements to calculate Unadjusted Actor Weights(UAW).

The details are given below as per Table 1, 2 and 3.

UUCP=UAW+UUCW=64+55=119
Adjusted UCP (AUCP) = UUCP*(.65+.01*TEF)
=119*(.65+.01*87) =180.88
Final Effort=AUCP*Conversion Factor
=180.88*13=2351.4

Taking 15% for project complexity and 10% for management, the results obtained were: Effort=2939.304 man hours =367 man days.

Table 1. Calculation of Unadjusted Actor Weight (UAW)

Actor	No. of use cases	Factor	UAW
Simple	0	1	0
Average	32	2	64
Complex	0	3	0
Total UAW			64

Table 2. Unadjusted Use Case Weights (UUCW)

Use Case	Type	Factor	UUCW
Simple	2	5	10
Average	1	10	10
Complex	1	15	15
Very Complex	1	20	20
Total UUCW			55

Table 3. Technical Factors

Factor	Assigned value	Weight	Extended Value
T1	5	3	15
T2	5	5	25
T3	2	1	2
T4	3	1	3
T5	3	2	6
T6	4	4	16
T7	2	1	2
T8	4	2	8
T9	5	2	10
Total			87

Actual effort = 390 man days.
Magnitude of Relative Error = | Actual Effort-Predicted effort |/Actual Effort * 100
=|390-367|/390*100 = 5.8%

PSO on UCP is applied by:

• Initializing the 10 particles, with 16 parameter values corresponding to actors, use cases, and technical factors in Use Case Point Analysis in the search space randomly. (In this case, an initial swarm of 10 particles was considered, as discussed earlier in proposed strategy. It may vary for different problems.)

The range of values for the parameters were bound in the intervals shown in Table 4. The initial values for the particles were taken randomly within this range.(Table 5)

Table 4. Range Taken For Weights Of Parameters

Parameters	Range
Actor	1-3
Use Cases	5-20
Technical Factors	1-5

Table 5. Initial Values of Some Particles

Parameter	Particle 1	Particle 2	Particle 3
Actor-simple	1	1	2
Actor-average	2	2	1
Actor-complex	3	2	3
Use Case-simple	5	6	10
Use Case-average	10	9	5
Use Case-complex	15	14	7
Use Case-very complex	20	19	18
T1	3	1	3
T2	5	2	2
T3	1	3	4
T4	1	4	1
T5	2	5	5
T6	4	3	3
T7	1	4	4
T8	2	2	1
T9	2	3	1

- Initial effort of all 10 particles is calculated by taking the effort equation as the objective function. This will be their initial local bests, and from these local bests, the best one is taken to be the initial global best.

  For particles, initial effort is obtained as: 3633.5, 2359.5, 3633.5, 3077.62 and so on

- In each iteration, all the 16 parameters for each particle are updated using the position and velocity equations for PSO. And then, objective function is again computed to find the local best and global best. Thus in each iterations, local and global best get updated, and the swarm moves towards optimality.

  In this problem 50 iterations are considered because results converge to an optimal solution within this limit. More iterations may be required, depending on the problem.

- Objective function is taken in such a way so as to maximize the effort but keeping it closer to the effort obtained by applying the basic method (UCP). So basic effort estimation equation of use case analysis is used.

- Finally after all iterations, results converge to 3032 man hours as total effort (including 25% for management and project complexity)= 379 man days

  Magnitude of Relative Error = | Actual Effort-Predicted effort |/Actual Effort * 100

$$=|390-379|/390*100 = 2.8\%$$

Example 2. Test Point Analysis
A project on “Random Number Generation Using Cellular Automata and LFSR.

Coupling" developed in C++ is taken for analysis. It has 11 modules and was tested in

2 months. Following are the parameters related to the project:

Total modules=11

Productivity Factor=0.7 (productivity factor is the time it takes for a tester to perform an activity, which depends on his skills and experience and is specific to an organization)

Now the well-defined variables under TPA are used for calculations:

$FDC_w = ((FI_w + UIN_w + I + C) / 20) X U$

Table 6. Calculation of FDCw

Module	FI	UIN	I	C	U	FDC_w
1	low	Low	low	Med	1	0.65
2	low	Med	low	Med	1	0.75
3	low	Med	low	Low	1	0.6
4	low	Med	Low	Low	1	0.6
5	med	Low	Med	Low	1	0.75
6	high	High	Low	Low	1	1.45
7	high	High	Low	Low	1	1.45
8	high	Med	High	High	1	1.8
9	med	Med	Med	Low	1	0.85
10	med	Med	Med	Low	1	0.85
11	high	Low	High	High	1	1.7

$QC_{dw} = \sum$(rating of QC/4) X weight factor of QC.

Table 7. Calculation of QC_{dw}

Module	Suitability	Security	Usability	Efficiency	QC_{dw}
1	Not imp	Rel unimp	Rel unimp	Rel unimp	0.187
2	Not imp	Rel unimp	Rel unimp	Rel unimp	0.187
3	Not imp	Rel unimp	Rel unimp	Rel unimp	0.187
4	Not imp	Rel unimp	Rel unimp	Rel unimp	0.187
5	Not imp	Rel unimp	Rel unimp	Rel unimp	0.187
6	Not imp	Rel unimp	Rel unimp	Rel unimp	0.187
7	Not imp	Rel unimp	Rel unimp	Rel unimp	0.187
8	Not imp	Rel unimp	Rel unimp	Rel unimp	0.187
9	Not imp	Rel unimp	Rel unimp	Rel unimp	0.187
10	Not imp	Rel unimp	Rel unimp	Rel unimp	0.187
11	Not imp	Rel unimp	Rel unimp	Rel unimp	0.187

Calculation of DTP:
Number of Dynamic Test Points (DTP) = FP X FDC_w X QC_{dw}

Table 8. Calculation of DTP

Module	FP	FDC_w	QC_{dw}	DTP
1	11.38	0.65	0.187	1.388
2	9.969	0.75	0.187	1.402
3	17.44	0.6	0.187	1.962
4	27.76	0.6	0.187	3.124
5	25.97	0.75	0.187	3.653
6	5.45	1.45	0.187	1.4827
7	5.54	1.45	0.187	1.4827
8	18.995	1.8	0.187	6.411
9	23.648	0.85	0.187	3.769
10	8.589	0.85	0.187	1.369
11	19.187	1.7	0.187	6.116
Total				32.1594

Calculation of STP (Static Test Points):
STP = FP X $\sum QC_{sw}$ / 500 = 16*0/500*23.64 = 0
Calculation of TTP (Total Test Points):
TTP = DTP+STP = 32.1594+0 = 32.1594
Calculation of Environment Factor:
Test Tools: No test tools used
Development Testing: Test plan Available
Test Basis: Documentation not developed according to standards
Development Environment: using old platform
Test Environment: Test platform is new
Test Ware: No Test Ware available
Environmental Factors = weights of (test tools+development testing+test basis+ development environment+testing environment+testware)/21=4+4+12+8+12/21 = 1.4

Calculation of Primary Test Hours:
PTH = TTP X Productivity Factor X Environmental Factor=32.1594*0.7*1.43= 32.19

Calculation of Total Test Hours:
Team size = 4
Planning and control tools: not available
Planning and Control allowance = weights of (team size + Planning and Control tools) X PTH = (3+6)*32.19 = 289.71

Total test hours TTH = PTH + Planning and Control Allowance
= 289.71+32.19 = 321.9 hrs = 40.2375 days

Actual testing time = 55 days

Magnitude of Relative Error = | Actual Effort-Predicted effort |/Actual Effort * 100
= (55-40.2375)/55*100 = 26.84%

Now PSO is applied on this technique by:

- Initializing swarm size of 10 particles, as discussed in proposed strategy.
- Random initial values to all parameters of TPA for the 10 particles are taken and total test hours are calculated, which is the initial local best. Some of the local best for particles were 485.78, 356.17, 474.357 , etc
- Initializing global best from this initial set of local best values.
- Applying iterations in which position and velocity of particles are updated each time, using the equations of PSO, resulting in new local and global best.
- Finally, after 50 iterations, results converge to total test hours=472 hour= 59 days (with 8 hours/day)

Magnitude of Relative Error= | Actual Effort-Predicted effort |/Actual Effort * 100
=|55-59|/55*100 = 7.2%

8 Comparison with Existing Approach

From the above case studies, the following results were obtained and list is in Table 9, Fig. 4 and 5:

Table 9. Comparative Results

Technique	MRE(without PSO)	MRE(after applying PSO)
Use Case Points (Example1)	5.8%	2.8%
Test Point Analysis (Example2)	26.84%	7.2%

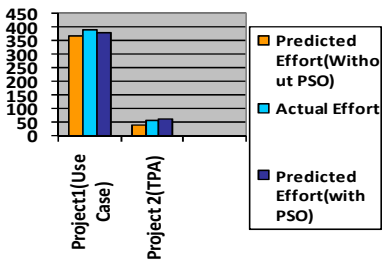

Fig. 4. Bar Graph for Comparing Effort

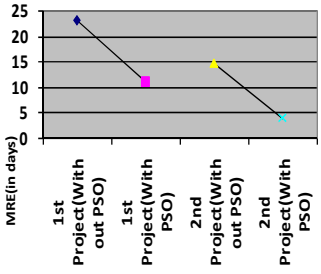

Fig. 5. Graph for Comparing Reduction in MRE

The results show that after applying PSO, the magnitude of relative error decreases, thereby making the predicted results closer to the actual ones.

9 Conclusion and Future Work

In this project, Particle Swarm Optimization (PSO) was applied on two techniques: use case points (UCP) and test point analysis (TPA) and the results led us to the conclusion that test effort estimation can be optimized by applying PSO. The results were compared with those obtained from existing methods, and were found to be closer to the actual effort.

The work done can be extended to optimize other effort estimation techniques. In this paper, only optimization of test effort estimation has been considered. PSO optimization can also be applied for estimating effort of software development. Possibility of further optimization can be explored by applying other variants of PSO.

References

1. Sommerville, I.: Software Engineering. Pearson Edition, India (2009)
2. Jalote, P.: An Integrated Approach to Software Engineering. Springer Science+Business Media, Inc., New York (2005)
3. Clarke, J., et al.: The Application of Metaheuristic Search Techniques to Problems in Software Engineering. SEMINAL-TR-01-2000 (2000)
4. Ferrucci, F., Gravino, C., Oliveto, R., Sarro, F.: Using tabu search to estimate software development effort. In: Abran, A., Braungarten, R., Dumke, R.R., Cuadrado-Gallego, J.J., Brunekreef, J. (eds.) IWSM 2009. LNCS, vol. 5891, pp. 307–320. Springer, Heidelberg (2009)
5. Sandhu, P.S., et al.: Software Effort Estimation Using Soft Computing Techniques. In: World Academy of Science, Engineering and Technology, pp. 488–491 (2008)
6. Nageswaran, S.: Test Effort Estimation Using Use Case Points. In: 14th International Software/Internet Quality Week, San Francisco (2001)
7. van Veenendaal, E.P.W.M., Dekkers, T.: Test Point Analysis: A Method for Test Estimation. In: ESCOM 1999 (1999)
8. Kaur, J., et al.: Neural Network-A Novel Technique for Software Effort Estimation. International Journal of Computer Theory and Engineering 46, 485–487 (2008)
9. Martin, C.L., et al.: Software Development Effort Estimation Using Fuzzy Logic: A Case Study. In: 6th Mexican International Conference on Computer Science (ENC 2005), Mexico, pp. 113–120 (2005)
10. The PSO website, `http://www.swarmintelligence.org/`
11. Parsopoulos, K.E., Vrahatis, M.N.: Particle Swarm Optimization and Intelligence: Advances and Applications. Information Science Reference, New York (2010)
12. Clemmons, R.K.: Project Estimation With Use Case Points. CrossTalk – The Journal of Defense Software Engineering, 18–22 (2006)
13. Chauhan, N.: Software Testing – Principles and Practices, pp. 335–340. Oxford University Press, New Delhi (2010)
14. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: IEEE Conference on Neural Networks, Piscataway, NJ, pp. 1942–1948 (1995)
15. Shi, Y., Eberhart, R.C.: A modified particle swarm optimizer. In: IEEE International Conference on Evolutionary Computation, pp. 69–73 (1998)