Volume 3, Issue 12, December 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Particle Swarm Optimization Approach for Component Based Software Architecture

Y.Mohana Roopa
Department of CSE, AITS, Tirupati
India

Dr. A.Rama Mohan Reddy
Department of CSE, SVU, Tirupati
India

Abstract— A component is an independently deliverable set of reusable services. Component-Based System (CBS) is a promised approach to build applications from deployed components. It provides efficiency, reliability, maintainability. Interpreting the results of performance analysis and generating an alternative design to build system from components is a great challenge in the software performance domain. To solve this problem genetic algorithms are used. The proposed Particle swarm optimization approach is used effectively to generate alternative design options in spanned design space and facilitate the design decision during the development process for component based software systems.

Keywords— Component based system, Software architecture, particle swarm optimization, genetic algorithm, software performance.

I. Introduction

Component technology has become a central focus of software engineering in research and development due to its great success in market. Reusability is a key factor that contributes to this success. With component technology, software systems are built by assembling components that have already been developed earlier, with integration in mind. With software component frameworks, the non functional codes are automatically generated, and system developers can focus on core business logic parts, without wasting time with common non-functional parts. The reuse of components and developers' focusing on core parts lead to a shortening of software development cycles and savings in software development costs. Furthermore, by versioning individual components and reconfiguring systems, the evolution of systems can be controlled, which makes software systems easy to maintain, dynamically upgrade and extend.

Today Component Based Software Development (CBSD) is getting accepted in industry as a new effective development paradigm. It emphasizes the design & construction of software system using reusable components. CBSD is capable of reducing development costs and improving the reliability of an entire software system using components. The major advantages of CBSD are low cost, in-time and high quality solutions. Higher productivity, flexibility & quality through reusability, replace ability, efficient maintainability, and scalability are some additional benefits of CBSD. In a recent survey conducted on component based software development from 118 companies from around the world, it is found that around 53% of the organizations are using component-based approach in its development [2]. If there are a number of components available, it becomes necessary to devise some software metrics to qualify the various characteristics of components. Software metrics are intended to measure software quality characteristics quantitatively. Among several quality characteristics, the reusability is particularly important. It is necessary to measure the reusability of components in order to realize the reuse of components effectively.

II. Genetic Algorithm

Assume an optimization problem contains at least one solution and the searching space is finite. Then, Genetic Algorithm (GA) [4] is able to locate the optimal solutions within the searching space. The basic unit of GA is chromosome or candidate. They are parameterized as a list of numbers which are the features representing the chromosome. In the optimization problems, there are some parameters to be optimized and a set of parameters forms one candidate. A few candidates are generated randomly and put in a pool. They are evaluated with the fitness function, or cost function. The fittest group of candidates always survives in the pool and they are mated as the parents for next generation. These parents form some pairs and born the offspring by the crossover and mutation operation [8,9]. The better candidates, supposedly, are evolved from the competitions among the candidates in last generation. After several generations, the combinations remained are the elites and the final solution is the best inside the pool.

III. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality PSO originated from the simulation of social behaviour of birds in a flock. In PSO, each particle flies in the search space with a velocity adjusted by its own flying memory and its companion's flying experience. Each particle has its objective function value which is decided by a fitness function. PSO is an evolutionary computation technique which is very similar to that of the Genetic Algorithm [5] where a particular system is initialized by a population of random solutions. In PSO along with each potential solution, randomized velocity is also assigned which constitute a particle. Each particle follows its coordinates in the problem space in connection with the best solution. Here the fitness value is also considered for the further process. This fitness value is referred to as *pbest*. The location of these solutions is regarded as *gbest*. In our proposed technique we have utilized a modified version of PSO. In this PSO we have assigned worst case as well along with the best case and also cross over operation is also included after the fitness selection which would further increase the possibility of selecting the best particle.

The PSO thus provides better solution and the steps involved are given in the below section.

A. *Steps in Particle Swarm Optimization*:

The various steps involved for implementing the PSO is explained below,

i .First initialize a population of particles (solutions) with position and velocity chosen randomly for n-variable in the Problem space.

ii. For each of these randomly generated particles evaluate the optimization fitness functions in n- variables.

iii. Now compare this fitness value with the particles *pbest* value. If these current fitness value is better than the *pbest* then chose the current fitness value as the pbest for the further processing.

iv. These fitness values is compared with the overall best previous values and if the current value is better than update the *gbest* for the current particles array index and value as the new *gbest*.

v. Change the velocity and the position of the particle and then repeat the steps until the criterion of better fitness is obtained. The velocity and the position of the particle are varied with the help of the below equations,

$$v_j(n+1) = v_j(n) + a_1 d_1 (g_j(n) - h_j(n)) + a_2 d_2 (g_j^{'}(n) - h_j(n)) \quad (1)$$

$$h_j(n+1) = h_j(n) + v_j(n+1) \quad (2)$$

vi. The process is repeated until the solution with better fitness value is obtained.

In the above equations, and represents the acceleration constants that is needed for combining each particle with the *pbest* and *gbest*. Updating the best position of the particle can be given as per the equation below,

$$g_j(n+1) = \begin{cases} g_j(n), & k(h_j(n+1)) \geq k(g_j(n)) \\ h_j(n+1), & k(h_j(n+1)) < k(g_j(n)) \end{cases} \quad (3)$$

The particle velocity at each dimensions are limited to the interval [$\pm V_{max}$] are then measured and is compared with the V_{max} .The V_{max} is an important parameter. The V_{max} helps in determining the resolution with which the region between present position and the target position are searched. Depending on the V_{max} values the particles are considered to provide a better solution. The above equations are utilized to calculate fitness of the solution and to select the better solution based on these fitness values.

Hence PSO can be utilized to solve the optimization problems just like other evolutionary algorithms. It can be applied in various fields like signal processing, robotics, simulations applications etc. Here we have employed PSO in order to select the better solution for the test case optimization. The various test cases are processed in the algorithm to obtain the best result suited for software architecture. The summarised PSO approach as shown in the Fig1.

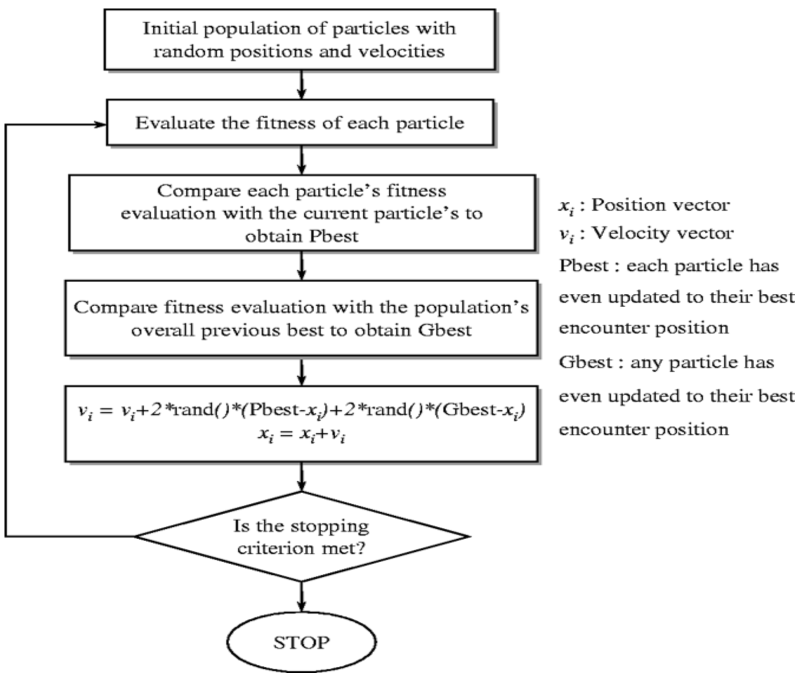
Initial population of particles with random positions and velocities
Evaluate the fitness of each particle
Compare each particle's fitness evaluation with the current particle's to obtain Pbest
Compare fitness evaluation with the population's overall previous best to obtain Gbest
$v_i = v_i + 2 * rand() * (Pbest - x_i) + 2 * rand() * (Gbest - x_i)$
$x_i = x_i + v_i$
Is the stopping criterion met?
STOP
x_i : Position vector
v_i : Velocity vector
Pbest : each particle has even updated to their best encounter position
Gbest : any particle has even updated to their best encounter position

Fig. 1: Flow chart of PSO

IV. RELATED WORK

The concept of Software Architecture (SA), which was matured in late 1980, has played an essential role in evaluating the design of complex systems [1]. Since then a broad set of notations, tools, and techniques have been offered, therefore a genuine base for designing enterprise and complicated system was established [4]. However, SA is a suitable stage to cope with software qualities. Notable efforts have been committed to facilitate the evaluation of Software quality in the level of SA [3]. Software architects are employing Model-Driven Development (MDD) [6] to manage architectural models of the system under development. The architectural models transformed into simulation-based or analytical models (e.g. Stochastic process Algebra, QN, Petri nets). Then, the probable problems are derived from the resulted model. However, the interpreting of the results of performance analysis is quite essential in the software performance prediction. Besides, the analysis of software alternatives results is still lacks of automation and is based on the skills and experience of analysts [7]. Furthermore, although, many approaches have been proposed and were successfully applied to predict software performance, still span of design space is serious problem when attempting to select the best design alternative. Meta-heuristics such as Genetic Algorithms (GAs) methods have proven its usefulness to solve the problem even with multi-degree of freedom. In recent investigations, PSO, an alternative search technique, often performed better then GA when applied to various problems. It describe performance prediction approach based on PSO for component-Based system development. The proposed approach aids developers to effectively trades-off between architectural designs alternatives. PSO can be used to provoke more efficient results. Outlines of this approach are presented and a case applied using GA is described to be applied to our approach in order to compare between the two techniques.

V. RESULTS

From the resulted set of solution, the set of optimal architecture configurations feasible with respect to the quality requirements is presented. Consequently, the software architect makes the trade-off decision and chooses one of the solutions. In our experiment, we have utilized the Particle swarm optimization for the Component Based software architecture. The implementation is done in the JAVA platform and the results are given as follows. In our proposed method we have utilized the Banking system as the source software.

Based on the fitness value of the parameter chosen the whole process labours in Particle Swarm optimization. For the additional processing the parameter with high fitness value is chosen. For both Particle Swarm optimization and Genetic Algorithm the fitness values of the chromosomes are computed for dissimilar iteration and the results are charted. By seeing the Table I the fitness value for the suggested technique verified to be superior to the technique where GA is applied.

Table I: Fitness value for Adaptive GA and GA for each iteration.

Iterations	Fitness value	
	Proposed method using PSO	Using GA
5	15	16
10	13	15
15	13	13
20	12	13
25	12	13

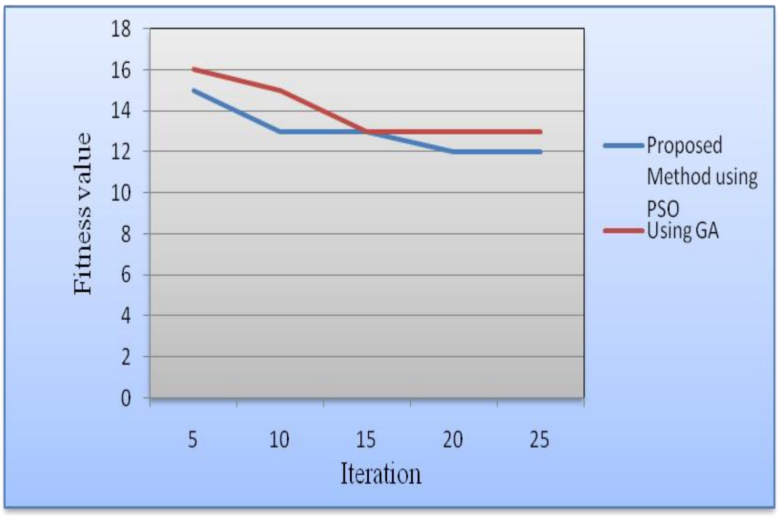

Fig. 2: Comparison of Fitness value between PSO and conventional GA

Based on the above table, the graphs are plotted and from the graph shown in fig 2, it is clear that the fitness values of PSO converges quickly when compared to GA. The computational time is well thought-out as the most important issue in software architecture as made cleared in the preceding section. The computational time for the software architecture plan based on the chosen component values(test cases) by means of both the PSO and GA are after that computed and the resulting values are charted. The significances of the computation time we attained for different test cases are shown in Table II.

Table II: Fitness value for PSO and GA for each iteration

Optimal Test Cases	Computational time	
	PSO	GA
5	804	4329
10	1122	4531
15	1781	4923
20	2078	4986
25	2603	5083

The graph is designed for computational time for dissimilar test cases based on the values shown in above table. Using PSO and GA the graphical representation of computational time for our suggested method was shown in Fig. 3. As shown in the graph, the computational time for PSO has been less significant when match up to that of GA.

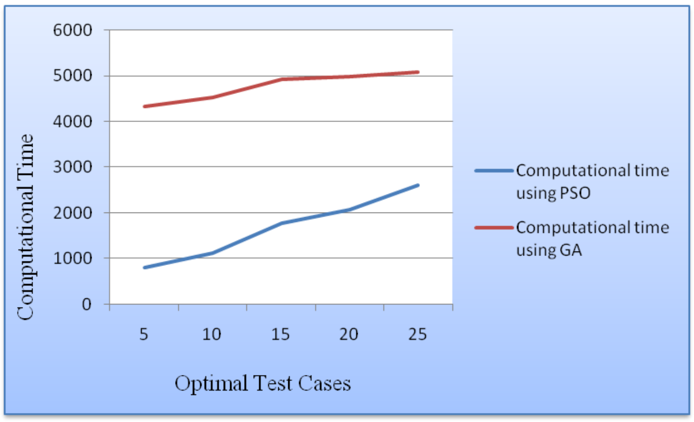

Fig 3: Comparison of Computational Time for PSO and GA

VI. Conclusion

Designing architectures that provide good quality criteria is quite difficult. Most of the proposed approaches do not provide substitution solutions rather they just recognize that performance is not satisfied. The architect needs to map the result back in the design model and manually develop new alternative. PSO has proved to be an efficient optimization method for optimization problems in different areas. PSO as search technique could be used to generate alternatives design automatically so it will contribute in improving the accuracy and decreasing the time of development. The use of PSO in the field of performance prediction will support the creation and evaluation of new architecture candidates. Therefore, a larger search space is explored and architects could be able to select between options that could improve the architecture of system. The improvement of architecture encompasses faster and more reliable software components. Our ongoing research aims to propose Particle Swarm Optimization (PSO) approach to support the right design decisions in developing software systems that have conflicting quality attributes. In this paper computational time is considered, more quality attributes planned to be added such as availability. Furthermore, expected discount in the cost as the number of requested components increased will be considered as well.

References

[1] Adil A. Aziz, Wan M. N. Wan Kadir and Adil Yousif, "An Architecture-based Approach to Support Alternative Design Decision in Component-Based System: A Case Study from Information System Domain, International Journal of Advanced Science and Technology Vol. 38, January, 2012, PP1-14.
[2] IEEE, Standard Glossary of Software Engineering Terminology, (1991), pp 610.12-1990.
[3] Dewayne E. Perry ,Alexander L. Wolf "Foundations for the study of software architecture"in ACM SIGSOFT Software Engineering Notes Homepage archiveVolume 17 Issue 4, Oct. 1992Pages 40 - 52

[4] Harman, M. “The Current State and Future of Search Based Software Engineering. in Future of Software Engineering”, 2007. FOSE '07. (2007).

[5] Kennedy, J. and R.C. Eberhart. A discrete binary version of the particle swarm algorithm. in Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., (1997), IEEE International Conference on (1997).

[6] hoshgoftaar, T.M., L. Yi, and N. Seliya, “A multi objective module-order model for software quality enhancement. Evolutionary Computation”, IEEE Transactions on, (2004), 8(6): p. 593-608.

[7] A. Martens, D.A. PerOpteryx/Hybrid Optimization Case Study. [cited (2011)9-11-2011]Available from: https://sdqweb.ipd.kit.edu/wiki/PerOpteryx/Hybrid.

[8] M. Mitchell. An Introduction to Genetic Algorithms. The MIT Press, Cambridge, Massachusetts, 1997.

[9] D. E. Goldberg. Genetic Algorithms in Serach, Optimization and Machine Learning. Addison Wesley, New York,1989 .