Software Project Scheduling Management by Particle Swarm Optimization

Article · December 2014

CITATIONS
2

READS
1,713

2 authors:

Dinesh Bhagwan Hanchate
Vidya Pratishthan's, College of Engineering, Baramati
122 PUBLICATIONS **216** CITATIONS
SEE PROFILE

Rajan Bichkar
G. H. Raisoni College of Engg & Mgmt, Pune
36 PUBLICATIONS **414** CITATIONS
SEE PROFILE

Software Project Scheduling Management by Particle Swarm Optimization

Dinesh B. Hanchate

Department of Computer Engineering (PG),

Vidya Pratishthan's College Of Engg.,

Baramati, Pune-413133, Maharashtra, INDIA

dinesh_b_hanchate[at]rediffmail[dot]com

Rajankumar S. Bichkar

Professor, E & TC,

G.H. Raisoni College of Engineering & Management (GHRCEM),

Wagholi, Pune-412207, Maharashtra, INDIA.

Rajankumar[dot]bichkar[at]raisoni[dot]net

Abstract: *PSO (Particle Swarm Optimization) is, like GA, a heuristic global optimization method based on swarm intelligence. In this paper, we present a particle swarm optimization algorithm to solve software project scheduling problem. PSO itself inherits very efficient local search method to find the near optimal and best-known solutions for all instances given as inputs required for SPSM (Software Project Scheduling Management). At last, this paper imparts PSO and research situation with SPSM. The effect of PSO parameter on project cost and time is studied and some better results in terms of minimum SCE (Software Cost Estimation) and time as compared to GA and ACO are obtained.*

Keywords: *PSO (Particle Swarm Optimization), SPM (Software Project Management), SPSM (Software Project Scheduling Management), SCE (Software Cost Estimation), SPE (Software Project Estimation).*

JEL: *-.*

Introduction

SPM is the most important and toughest job in software engineering organizations. For successful project management it needs the proper planning and scheduling of each software development activity. Software planning needs proper requirement and estimation of software for fulfilling those requirements. Project management is vital to the effective application of organizational resources to competing demands within and across projects. The effective application of project management, however, is predicated upon accurate estimates of the project budget and schedule.

Software engineering measurement and analysis specifically, cost estimates and its scheduling management initiatives have been at the center of attention for many more industries. Software managers come across usually with software projects that contain errors or inconsistencies and that exceed budget and time limits, too. A general assumption in the software engineering is that, software development is a process that is subject to inherent laws.

Organizations must decide how to allocate the available resources based on predictions of the unknown future. Accuracy in prediction of effort development, effort can reduce the cost from inaccurate estimation, misleading tendering trades and disabling the monitoring progress. Accurate modeling can assist in scheduling resources and in the evaluation of risk factors. Planning and scheduling are usual things and common to dif-

ferent engineering domains. Whether the project is small or huge, both planning and scheduling are equally important. Planning and scheduling are somewhat different activities. The plan defines what has to be done. It restricts on how to do it.

The schedule imparts specifications for how and when it will be committed. The plan, not only, refers to the time estimation and decides the resource for each activity, as well as the precedence and dependence between activities and other constraints. The schedule always point out the temporal assignments of tasks and activities which are to be executed in the plan. In addition, somehow all the project consists a set of objectives utilized to measure the performance of the schedule and/or the plan feasibility in practical. The project objectives give the overall performance of the plan and schedule. Scheduling does require the integration of different data types and kinds.

Constructing a schedule takes into accounts models of processes, roles and relationships between tasks and resources, involvement of objectives and performance measures. It makes a bond of an underlying data structures and algorithms that make them a tie and all together. Schedules assign resources to tasks and tasks to resources as required and given specific times. Tasks may be anything from managerial task to modularization of software assigned to people as a resource. Resources include 4Ps in SPM and these are people, process, product and project. The typical objectives of scheduling include minimizing the duration of the project, maximizing the net present quality of the project, or minimizing the number of products that are delivered late.

Since 1950s, scheduling and planning methods have been analyzed and put forward. Even though the methods are for finding optimal solutions to some of the specific scheduling problems and its formulations, somehow many methods do not work properly when constraints, its structure and/or objectives get modified. Many methods do not give the good performance when faced with problems of significant size. In some

cases, simply finding the feasible solutions become a considerable challenge. In general and, usually, scheduling problems are NP-hard.

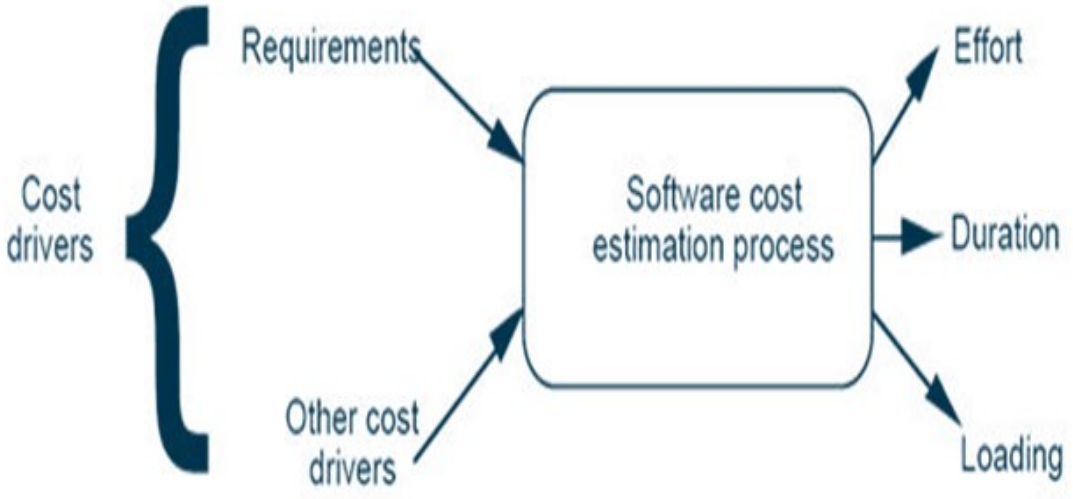

Figure 1 — Classical view of software estimation process

(Vigder and Kark, 1994)

It means there are no known algorithms for finding optimal solutions in polynomial time. But, there are algorithms for solving some forms of the problem, but they, mostly and typically, take so long (i.e. more than polynomial time) when the size of problem size grows or when other more constraints are added. As a result, most research has worked on either simplifying the scheduling problem to the point where some algorithms can go up to the solutions, or to machining heuristics for finding optimum and good solutions. Genetic algorithm gives a good solution for the complex, combinatorial nature of most scheduling problems. That has led many researchers to experiment with GAs. But, We will discuss PSO and it's factors effect on SPM.

Software Project and Cost Estimation

SPE involves measurement and analysis of the project in terms of project size, effort and cost. SCE is the prediction of hours of work and the number of workers needed to develop a project. Fig.1 gives a brief classical view of SCE. The cost drivers [1] and other drivers are taken as input to the user requirements. As shown in Figure 1, the software requirements for the projects and given by clients are the prim and basic input to the process which forms plinth for SCE [2]. The cost estimation is used to be adjusted as per a number of other cost drivers [3] to arrive at the baseline for EAC (Estimate at Completion) [4]. Cost drivers are usually used as prototype methodology, software professional skill-levels, project risk analysis, personal factors as an experience, system complexity and programming language. Classical SCE gives the effort, duration (schedule) and loading on the people of 4Ps [5]. Manpower loading imparts a number of personnel (which also includes management personnel) that are allocated to the project with respect to the function of time. Project duration (or sometimes called as schedule) is needed to complete the project. Effort is amount of team work with quality productivity needed to accomplish the given project in required duration.

SCE techniques fall into three general categories which includes algorithmic models, expert judgment and Machine Learning (ML) techniques. Among algorithmic models, COCOMO (Constructive Cost Model) is the most commonly used Algorithmic cost modeling technique because of its simplicity for estimating the effort in person-months for a project at different stages. COCOMO uses mathematical formulae to predict project cost estimation. COCOMO is taken as a base model for the software cost estimation. Expert judgment is a non-structured process in which experts make decisions to the technical problems based on knowledge and experience in order to give accurate results than other techniques and doesn't require any previous historical data [6] [7] [8]. And the third technique is ML by using the neural networks. Neural networks have been found as

one of the best techniques for software cost estimation [9]. Numerous researchers and scientists are constantly working on developing new software cost estimation techniques using neural networks [10].

In 1981, Boehm introduced COCOMO model for estimating effort, cost, and schedule for software projects. As he states, COCOMO is used in the course to explain a some current software engineering guidelines and their impact on the software life-cycle. A scatter plot of effort in man -months versus project size in delivering source instructions shows that the COCOMO data clusters into three distinct groups of effort (measured in man-months) versus product size (measured in delivering source instructions). Boehm named the three groupings the Organic, Semi detached, and Embedded modes. It gives the relationship between effort-size and cost as

$$MM = A_i \times (KDSI)^{B_i} \quad (1)$$

$$TDEV = C_i \times (MM)^{D_i} \quad (2)$$

where, A_i, B_i, C_i and D_i are the constants that differ for each of three modes; KDSI is thousands of Delivered Source Instructions; MM is effort in Man-Months; TDEV is development time in months. He then provided tables to calculate the distribution of effort among project phases and the distribution of effort for eight major project activities within each phase. He termed the effort and schedule equations for the three modes to be Basic COCOMO. Boehm, then presented Intermediate COCOMO. Based on literature research and the help of several experts, he used the Delphi technique to identify 15 cost drivers and their effort multiplier values to account for additional influences on the project.

The resulting equations are of the following form

$$MM = A_i \times (KDSI)^{B_i} \times \prod EM_j \quad (3)$$

where, Π EMj is the product of 15 Effort Multiplier Factor (EMF) values that vary within ranges around 1.0. Values, which are less than 1.0 indicating that the next project is estimated to require less effort and time than the projects from which the equations were derived and more than 1.0 to require more effort and time. COCOMO Cost Drivers are qualitative designators for EMFs. For the Intermediate version of COCOMO 81, Boehm also provided a Component Level Estimation Form (CLEF) and a procedure for developing and combining estimates for multiple software components of a large system. In addition, he showed how to compute an Adaptation Adjustment Factor (AAF) to determine equivalent new source lines of code for software to be adapted (reused). AAF uses three parameters

- Percent of Design Modified (DM),
- Percent of Code Modified (CM) and
- Percent of Integration Required (IM) for the code to be adapted.

Particle Swarm Optimization

PSO is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995, inspired by social behavior of bird flocking schooling. The algorithm is, universally, widely used and rapidly developed for easy implementation. PSO has now become popular as it has fewer particles required to be tuned [11]. The main idea of the PSO principle is presented in [12]; the advantages and the shortcomings are summarized in [12]. PSO has been successfully utilized in many and more development, research and application areas [13] [14], in the past several years. Another reason that PSO is attractive due to few parameters to be adjusted. PSO has been used for approaches that can be used across a wide range of applications, as well as for specific applications

focused on a specific requirement.

It is developed from swarm intelligence and is based on the research of bird and fish flock movement behavior [15]. For searching and collecting the food, the birds do scatter or go together before they find and finalize the place where they can get their food. During the searching for food from one place to another, there is often a bird that can sense and smell the food very well. Hence the bird, having better food intelligence resource information, is sensible perception of the place where the nutrition can be located and found. The birds will usually congregate to the place where food can be found at any time while relocating and locating the food from one place to another due to transition the information concerned for good food. As far as the PSO algorithm is concerned [16], solution swarm is synonymous and compatible with the bird swarm [17]. As the birds flocking from one place to another is equal to the development in swarm solution. The good information is considered to be equal to the most optimist solution, and the food resource is the most optimist solution during the whole course.

PSO [18] simulates the behaviors of bird flocking by following scenario. All birds in a group search food in an area. There will be only one piece of food in the selected area being searched. All the birds don't know about the food and where the food is. So what's the best strategy to find the food? The effective one is to go after the bird which is closest to the food. This learned scenario is used as PSO to solve the optimization problems [19]. Each single solution, in PSO, is a bird in the search space. We say it as a particle. All of the particles have fitness values which are evaluated by the fitness function, and have velocities which direct to the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO does not use the filtering operation (such as crossover and/or mutation) and the members (particles) the entire population are maintained through the search procedure. Each particle flocks around in the

multi-dimensional search space with its velocity. This Velocity is updated by experience of particle and its neighbors or by or the experience of the whole swarm [20] [21]. There are two variants in PSO namely local neighborhood and the global neighborhood. Each particle travels in the direction of the best of its previous position. This is called as gbest solution in the whole swarm. Another is the local variant, lbest, particle navigates to its best previous position and in the direction of the best particle in its restricted neighborhood [12]. Since PSO can be successfully applied to optimize various continuous nonlinear functions [15] [16] due to its some advantages including its simplicity and easy implementation. Therefore, this paper is to employ PSO on solving SPS (Software Project Scheduling) which is a typical combinatorial optimization problem.

PSO concept consists of, at each step, the velocity of (accelerating) each particle toward its best and best locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward pbest and lbest locations. Fig. 2 shows the PSO search space and the points [22]. It also shows the corresponding change in the velocity and position. After finding the two best values, the particle updates its velocity and positions by following equations.

$$v[i + 1] = \omega \times v[i] + PI + SI \quad (4)$$

$$PI = c1 \times r1 \times (p_{best}[i] - position[i]) \quad (5)$$

$$SI = c2 \times r2 \times (g_{best}[i] - postion[i]) \quad (6)$$

$$position[i + 1] = position[i] + v[i] \quad (7)$$

Where, PI is Personal Influence, SI is Social Influence, v[i+1] is updated velocity (path direction), position[i+1] is the current particle (solution). ω = inertia weight, V[i] =initial velocity, C1 = social parameter (weight of local information), C2 = cognitive parameter (weight of

global information), p_{best} = personal or local best (best position of the particle), g_{best} = global best (best position of the swarm), position[i] = initial Particle position, position[i+1]=updated particles position, r1, r2 = Fraction random values between 0 and 1 [18] [12] [23].

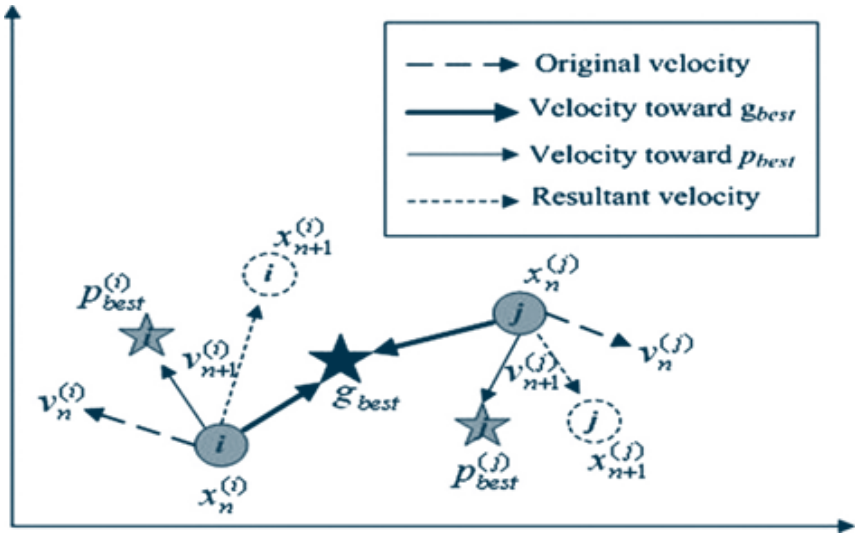

Figure 2 — PSO search space and points.

Some assumptions

Number of particles are, usually, kept between 10 and 50. C1, C2 values shows how much importance given to the personal best value and to neighborhood best value respectively. Usually, C1 + C2 = 4 (empirically chosen value [24]).

In PSO, multiple solutions, particles exist in the group. At every and each unit time, the particle flies towards its own best position and migrate to the best position among all particles. Each particle has its position and velocity information. This information is represented by the vectors. Each particle p has position vector P and with its velocity vector V and, is represented by $P_p = (P_{p1},..., P_{pD})$ and $V_p = (V_{p1}, ..., V_{pD})$, respectively, where (d =1, 2, ..., D) and (p = 1, 2, ..., N). D is the dimension of

the particle. It is also known as number of particle variables where, N is the number of particles in population

and $P_{pD} \in [P_{min}, P_{max}]$ [25].

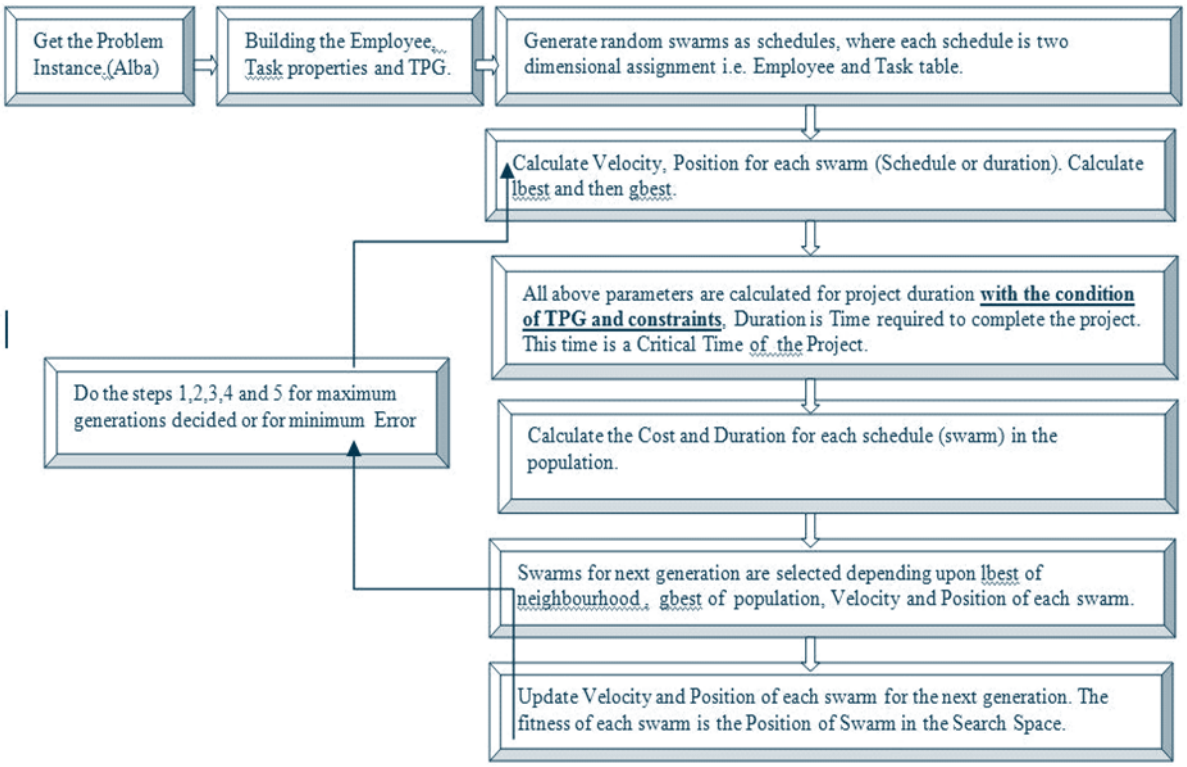

Figure 3 — Flow of SPM-PSO architecture.

PROBLEM DEFINITION

The software project scheduling problem is where the aim is to minimize the time and cost. The same is adopted in this paper by the authors from [26] and [27]. The some part of problem definition is taken as it is from the authors' paper [26] and [27]. The employees and task properties are taken from Table 1 and Table 2.

TABLE I — Employee Properties: 10T5E4S

Employees	Number of skills	Employee possessing Skills	Salary in $
Emp0	4	0, 8, 3, 1, *	11193
Emp1	4	1, 8 ,9 ,0 , *	11458
Emp2	5	4, 5, 8, 9, 3	9936
Emp3	5	5, 1, 6, 3, 0	9502
Emp4	4	2, 3, 5, 7, *	10448

TABLE II — Task Properties: 10T5E4S

Task	No. of Skills	Skills required for Task	Effort
Task0	2	6, 9, *	4
Task1	3	5, 9, 0	11
Task2	2	2, 6, *	7
Task3	3	2, 5, 3	10
Task4	3	2, 3, 6	7
Task5	2	5, 8, *	8
Task6	3	9, 1, 0	12
Task7	2	1, 3, *	4
Task8	2	5, 2, *	8
Task9	2	0, 2, *	5

A project schedule is an assignment of the tasks to the 4Ps at particular duration by considering all the constraints of the project to get the optimal solution with optimum cost and time. Each task requires a set of skills and effort.

Let T be a set of tasks, T= $\{T_i \mid i=0, ..., n-1\}$, where n is the number of tasks; E be a set of employees, E= $\{E_j \mid j=0,..., e-1\}$, where e is number employees; S be a set of skills, S= $\{S_k \mid k=0, ..., m-1\}$, where m is the total number of skills; ES be a set of skill of employees, ES= $\{ES_i \mid i=0, ..., m-1\}$; EF be the effort required for the tasks in T, EF= $\{EF_{Ti} \mid j=0, ..., n-1\}$, where EF_{Tj} is the effort required for task T_j.

The skills required by tasks are represented by an n × m sized task skill matrix i.e TS, where TS= $\{TS_{ik} \mid i=0, .., n, k=0, ..., m\}$

Each element TS_{ik} of task skill matrix TS is either 0 or 1, depending on whether task T_i requires skill S_k.

$$TS_{ik} = \begin{cases} 1 & \textit{if Task } T_i \textit{ requires skill } S_k \\ 0 & \text{otherwise.} \end{cases}$$

The employee skills are represented by an e × m sized task skill matrix i.e ES, where ES=-ES_{jk} | j=0, ..., n-1, k=0, ..., m-1}. Each element ES_{jk} of the employee skill matrix ES is either 0 or 1, depending on whether task E_j has S_k as

$$ES_{jk} = \begin{cases} 1 & \text{If Employee } E_j \text{ has skill } S_k \\ 0 & \text{otherwise.} \end{cases}$$

The dependence [28] between the tasks is given by task dependency matrix (TD) of size of $n_{\times}$ n. Its elements are given as,

$$TD_{ik} = \begin{cases} 1 & \text{If Task } T_i \text{ depends on } T_k \\ 0 & \text{otherwise.} \end{cases}$$

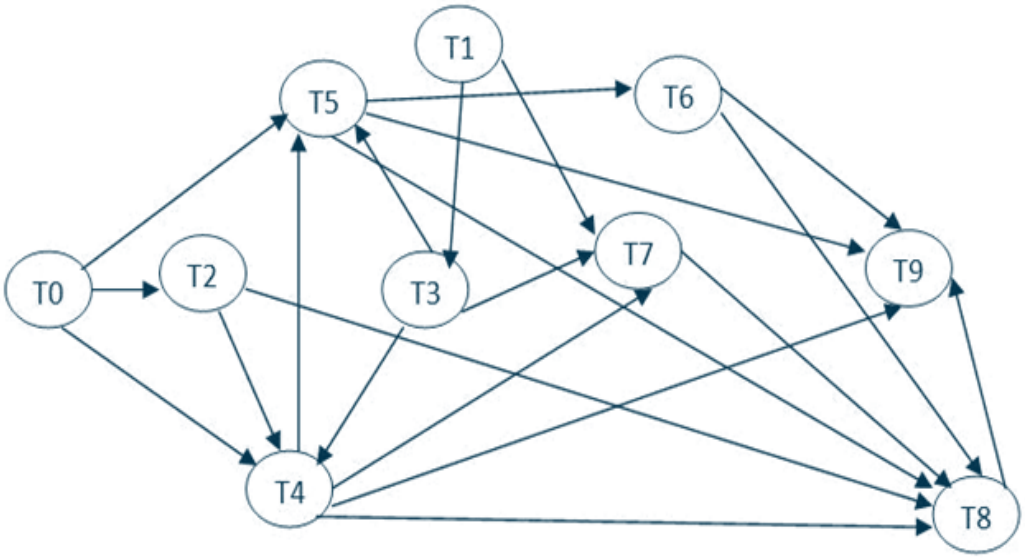

Figure 4 — TPG for 10 Tasks applied on 5 Employees who posses 4 to 5 skills

Finally, SHD is a n×e sized task assignment matrix of duration (in months) assigned to each employee on various tasks. The duration may be in years, months, quarters or weeks. TD matrix is obtained from the task precedence graph (TPG as shown in Figure 4). The Task Precedence Graph shows the precedence relation between the tasks, is an acyclic Graph, G(T, EG) where the T represents the set of all task nodes included in the project and EG is the set of edges between dependent tasks [29]

[28].

- During the assignment of employee to task following hard constraints must be satisfied.
- No employee must be overloaded i.e., during parallel execution its dedication should not go above 1.0. All skills must be matched.
- Select the solution whose salary and duration should be minimum.

SPS (SOFTWARE PROJECT SCHEDULING) PARTICLE SWARM OPTIMIZER (SPS-PSO)

The algorithm of SPS-PSO is almost same as the standard PSO.

Algorithm 2: SPS-PSO Algorithm

SPS-PSO step1st : Initialization. //
We have G, the number of generations as g = 1 in G, +1. In first generation at g=1, initialize randomly the schedule position P_{SHD} and its velocity V_{SHD} for each schedule SHD, and initialize P_{SHD} = (P_{SHD1}, ..., P_{SHDD}) with a copy of X_{SHD}. //

Evaluate the objective function f(X_{SHD}) for each schedule SHD.
Find P_g with the best function value among all the schedules. //SPS-PSO step2nd : Finding of best schedule and swarm in SPS-PSO.//
while { g = 0 to G } do
  //Find l_{best}.//
  for SHD=1:N do
    Evaluate the fitness f(P_{SHD})
    if f(X_{SHD}(g)) < f(P_{SHD}) then
      the personal best position (called l_{best}), P_{SHD} = X_{SHD}(g)
    end

  end

  //SHD_{best} is also called as l_{best} and here it is P_{SHD}. Find g_{best} i.e P_g with the best function value among the all schedules.//
  if P_{SHD} < P_g then the global best position (called g_{best}) P_g=P_{SHD} end

  //Finding of inertia, social influence, personal influence and the inertia weight are given bellow. Updating velocity V_{SHD} and position X_{SHD} of each schedule SHD is done at its l_{best} and g_{best} according to inertia weight at generation g. //

```
    //SPS-PSO step3rd : Finding of all the SPS-PSO parameters. I is an inertia.
    ω  determines how much of the previous velocity of the schedule is pre-
    served. //
    I=ω ×  V_SHD(g)

    PI=c1× r1×(P_SHD − X_SHD(g))

    SI=c2× r2× (P_g − X_SHD(g))

    V_SHD(g + 1)= I + PI + SI

    X_SHD(g + 1) = X_SHD(g) + V_SHD(g+1)

    g = g+1.
  end
```

The most important feature of SPS-PSO is that all particles are 2-dimensional schedules (as shown in the table-4). In other words, the schedules are connected to adjacent schedules (i.e. particles) by the neighbourhood project duration relation, which dictates the optimized assignment of employee to task. Furthermore, the particles which share the local best position particle between the neighbourhood particles are closely connected with the less difference in the duration. In our research, the neighbouring particles are defined as the particle with similar cost values or as the particles whose position are close to each other. Since, in this paper, the particles, which are directly connected in the flock which are closer to the duration of l_{best}, are defined as neighbourhoods [25].

RESULTS AND DISCUSSION

We implemented the SPS-PSO algorithm and the GA application to SPSP in Java (JDK 1.0.7) and run the all these experiments on an Dell Intel(R) Core(TM) i-5 2450M CPU @ 2.5 PC running Windows 7.0 ultimate. SPS-PSO ran every configuration le for 10 trials and we then keep quality of solutions by averaging the values. The instances are produced by the

instance producer to have configuration less [30]. All the data sets given by [30] have been experimented as test data sets and given a name corresponding to the task number and employee number. For example, 5e10t represents an example instance, which has 5 employees and 10 tasks. There are 5 groups of instances used in our experiments. The instances include 5e10t, 10e10t, 15e10t, 20e10t, 10e20t. Every group contains 4 instances with different number of employees and tasks. All groups of instances are used in the experiments of PSO parameter setting.

Table 3 shows a comparative chart of results for cost and duration obtained, i.e. ACO [31], ACO[30], PCO (OUR approach). Table 3 also shows the comparative results of AGA and Ting-Tang's approach which is taken from authors paper [26]. This shows that the PSO gives comparatively better result than AGA, SGA and ACO. We did a study of the all the parameters stated earlier by keeping one as variable and other constants. Figures 5 to 15 shows the different scenario for this scheduling output. As we increase the number of generations from 150 to 500 by keeping 50 generations gap, we can see in the Fig. 5 cost goes down and there is improvement in the results in fig. 7 where we kept C1 and C2 equal to 2 and ω =0.5 and population is kept 20.

As population size changes there is a change in the cost also but we got hit ratio nearly 70%. Fig.7, 12, 16 shows the linear changes in the output parameter Cost, Duration and fitness of schedules with respect to weight. Fig.16 is approximately opposite to the Fig. 7 and 12 as the cost and duration are inversely proportion to fitness.

For various values of C1 and C2 where it's addition must be 4, we can find that as local exploitation goes on increasing then also the search space gives results in duration in Cost and duration. But, C1=C2=2 gives good optimum results. Usually and generally, the number of particles is kept between 20 to 50 for better results. We kept on changing for the only 3 sizes 20, 30 and 40 which gives the conclusion that if we keep on

increasing the size of the population the results gives better and better solution.

TABLE III — Comparative chart of results for cost and duration obtained by : KNV-ACO, ALBA-ACO, SPS-PSO(OUR approach)

Sr. No.	EMP	TASK	Skills	TYPE	DURA-TION	SCE in $
1	5	10	2 to 3	KNV-ACO	24.62	959910
2	5	10	2 to 3	ALBA-ACO	23.62	1209520
3	5	10	2 to 3	SPS-PSO	23.03	947780
4	10	10	2 to 3	KNV-ACO	16.20	981023
5	10	10	2 to 3	ALBA-ACO	15.23	1220530
6	10	10	2 to 3	SPS-PSO	16.66	924567
7	15	10	2 to 3	KNV-ACO	8.03	957209
8	15	10	2 to 3	ALBA-ACO	8.23	1225200
9	15	10	2 to 3	SPS-PSO	8.60	734112
10	10	20	2 to 3	ALBA-ACO	51.30	2400000
11	10	20	2 to 3	SPS-PSO	48.72	1941030
12	15	10	4 to 5	KNV-ACO	9.41	959910
13	15	10	4 to 5	ALBA-ACO	8.23	1212790
14	15	10	4 to 5	SPS-PSO	9.24	1165662
15	15	10	6 to7	KNV-ACO	8.03	957209
16	15	10	6 to7	ALBA-ACO	8.87	1225200
17	15	10	6 to7	SPS-PSO	8.76	1101411

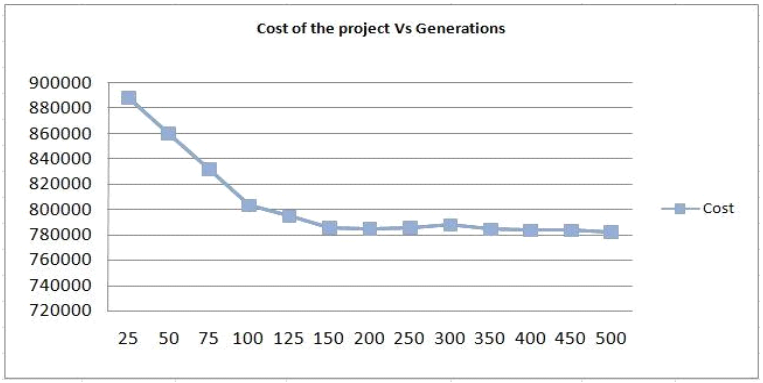

Figure 5 — SCE Vs Generation: 10T5E4S

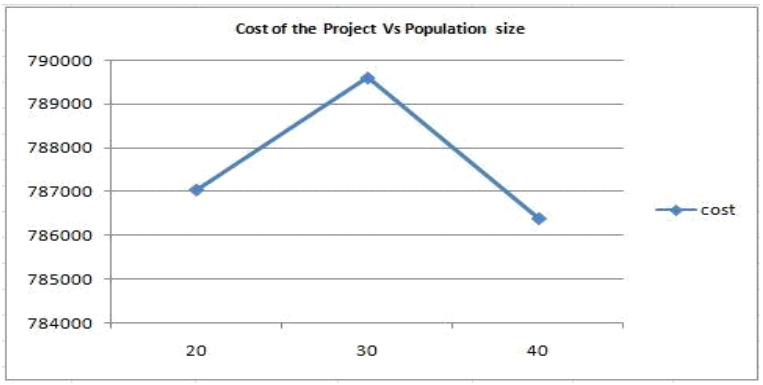

Figure 6 — SCE Vs Population Size: 10T5E4S

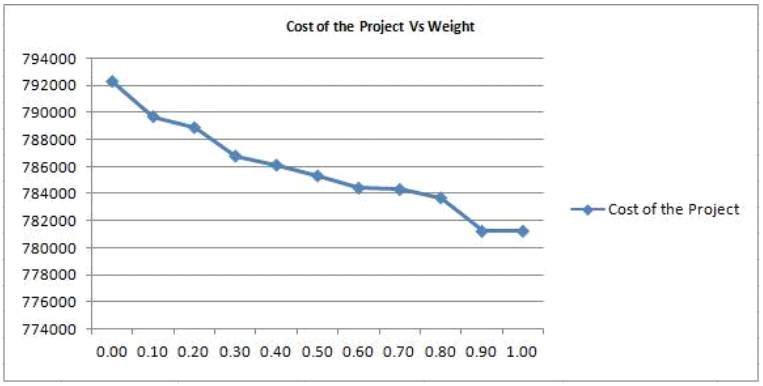

Figure 7 — Swarm size =20; Max. iteration = 100;C1=2;C2=2; ω= 0 to 1.0 with + 0.1 increment, Cost Vs Weight: 10T5E4S

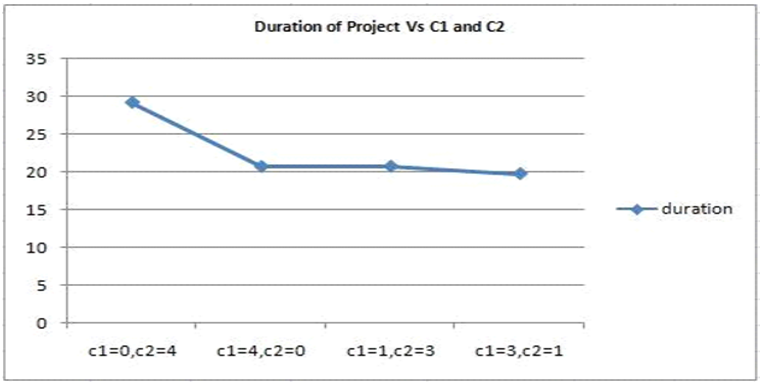

Figure 8 — SCE Vs C1 and C2:10T5E4S

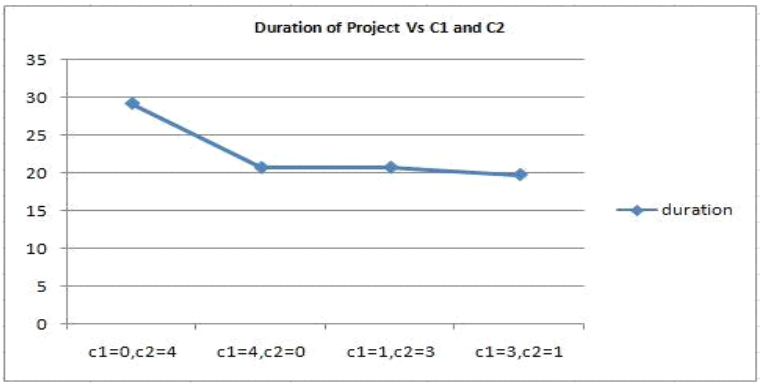

Figure 9 — Duration Vs C1 and C2: 10T5E4S

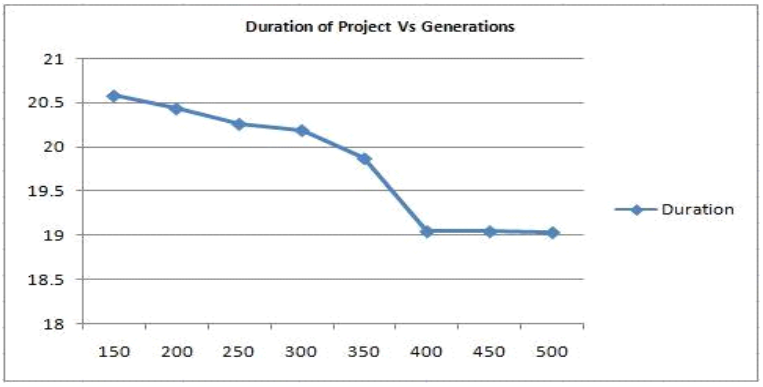

Figure 10 — Swarm size= 20, Max iteration is in between 150 to 500, 50 C1= 2; C2= 2; ω = 0.5; Duration Vs Generations: 10T5E4S

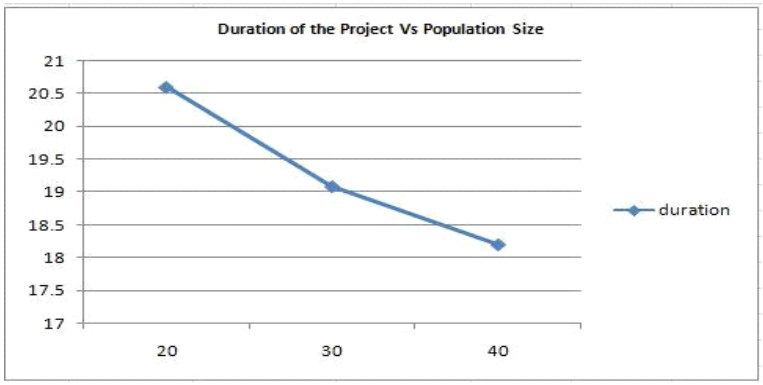

Figure 11 — Duration Vs Population Size: 10T5E4S

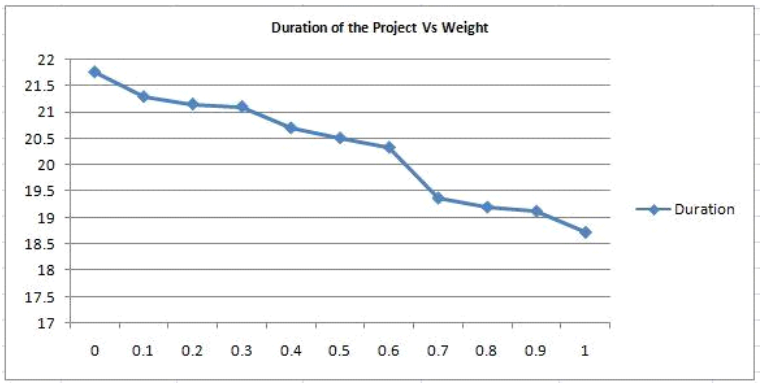

Figure 12 — Duration Vs Weight: 10T5E4S

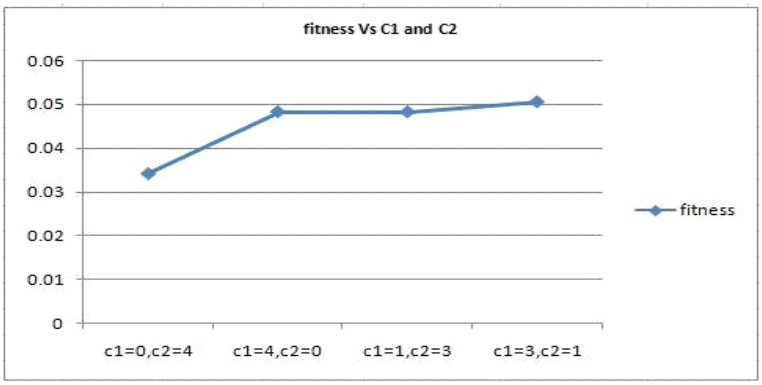

Figure 13 — Swarm size = 20; Max. iteration =200; C1 varies; C2 varies; ω = 0.5; Fitness Vs C1 and C2: 10T5E4S

Figure 14 — Fitness Vs Generation: 10T5E4S

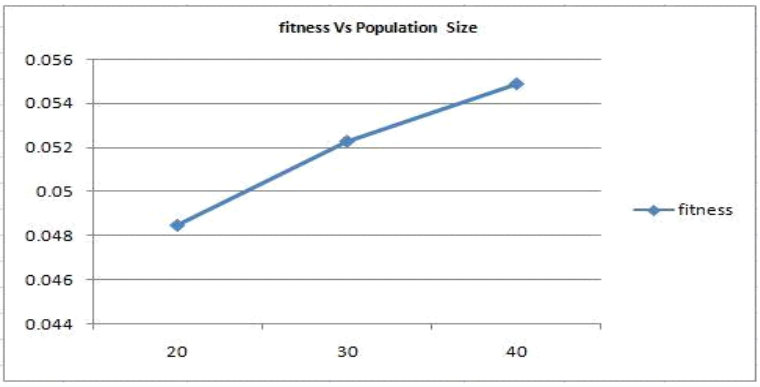

Figure 15 — Max. iteration = 200; C1 2; C2 2; ω = 0.5; Fitness Vs Population Size: 10T5E4S

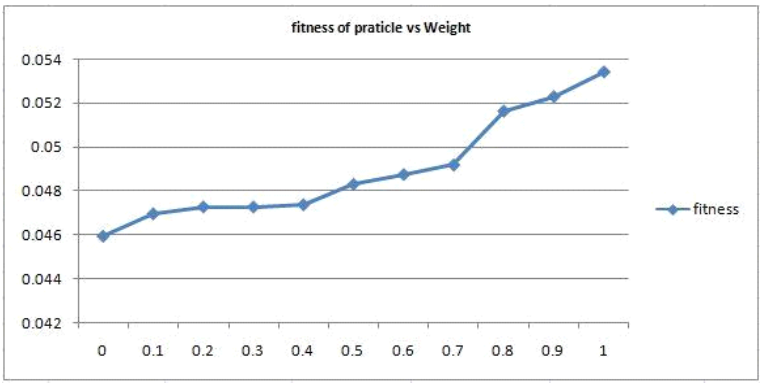

Figure 16 — Fitness Vs Weight: 10T5E4S

Heuristic Information

All the output tables with results of hit rate is experimented by groups with different heuristics [32]. It is observed that heuristic information has a distinct influence on the hit rate of solutions. Among the C1, C2, ω , l_{best}, G_{best} heuristic information, ω is the best after comparing the hit rate and fitness values of feasible solutions. In our experiment, we have taken 20 particles as it is usual practice in the PSO literature to limit the number of particles in the range 20 to 60 [33] [34].

TABLE IV — Dedication of employee for the tasks (Optimum SPS-PSO Positional-schedule value at g+1=G)

	E0	E1	E2	E3	E4
T0	0	1	1	1	0
T1	1	0.5	0.5	0.5	1
T2	0	0	0	1	1
T3	1	0	1	0.5	0.5
T4	1	0	1	1	1
T5	1	1	1	1	1
T6	1	1	1	1	0
T7	0.5	0.5	0.5	0.5	0.5
T8	0	0	1	1	1
T9	1	1	0	1	1

TABLE V — Dedication of employee for the tasks (Optimum SPS-PSO Velocity-schedule value at g+1=G)

	E0	E1	E2	E3	E4
T0	0.125	0	0	0.875	0.625
T1	0.5	0	0	0	0.75
T2	0	0.375	0	0	0
T3	0.125	0.75	0	0.125	0
T4	0	0	0	0.875	0
T5	0.625	0	0	0	0
T6	0.125	0.375	0	0.375	0.75
T7	0.5	0.125	0	0.25	0.625
T8	0.625	0	0	0.125	0
T9	0.875	0.875	0.625	0.375	0

Effect of Learning Factors on Software project scheduling: l_{best} and g_{best}

We found one neighbor is the g_{best} as all the particles are neighbors of each other and with g_{best} learning swarms converge fast (Global best PSO). However, in Local best PSO (l_{best}) only a K number of particles are neighbors. The swarm may converge slower, but the probability of locating the global optimum is more. It is a scheme for determining the neighbors of each particle. Ring and Star are two common neighborhood topologies. In ring particle is connected only to its immediate neighbor. The g_{best} is special case of a ring, called as star topology. We implemented star topology with V_{max} as maximum velocity Vp of each particle clamped

to V_{max} , which facilitates global movement as V_{max} becomes larger and larger. In other side, smaller value of V_{max} stick to local exploitation. C1 and C2 are acceleration constants. Initial version of PSO used values, C1 = C2 = 2. Best results are observed when we choose a larger cognitive parameter, C1, than a social parameter, C2, but with C1 + C2 = 4. The parameters r1 and r2 [35] are used to maintain the diversity of the population. They are uniformly distributed in the range [0, 1]. The inertia weight is usually used to develop to have control over the exploration and exploitation as to eliminate the need for V_{max}. In our experiments, inertia weight has provided improved performance. As initially developed, often is decreased linearly from about 0.0 to 0.9 during a run. A suitable selection of the inertia weight provides a balance between global and local exploration and exploitation [36] [37] [38]. The suitable ω =0. 5 is the best for getting better results rather than another value.

TABLE VI — Tasks and Software Cost Estimation in the last generation

Tasks	Cost in $
0	41194
1	116563
2	69825
3	103677
4	71887
5	84057
6	126263
7	42029
8	79695
9	53250
Total	788439

TABLE VII — Tasks and Duration in the last generation

Tasks	Duration
0	1.33
1	3.14
2	3.50
3	3.33
4	1.75
5	1.60
6	3.00
7	1.60
8	2.67
9	1.25
Total	16.91

Conclusions

In this work, the PSO optimization algorithm provides a much better solution as compared to the other optimization techniques such as ACO and GA. One can conclude by getting the previous researchers and our work that the PSO algorithm is used in many engineering applications for optimization of CSP problems and rather, it gives good output and performance than other optimization algorithms. The fitness function of minimizing the duration with cost is selected for the present work for equality and inequality constraint objective function. We did change the parameters such as inertia weight, social parameter C1, cognitive parameter C2 and the number of particles. It gives the best performance in between 0.5 to 0.8 inertia weight when both social and cognitive parameters are

given as 2. It is observed that if the velocity is too low, then algorithm goes too slow, if velocity is too high then algorithm too unstable. It is demonstrated that PSO gives better results in a faster and cheaper way compared with other methods.

References

[1] J. Crespo, E. Garca, and C. B. Km, "On aggregating second-level software estimation cost drivers: A usability cost estimation case study," 2004.

[2] S. B. Samuel Lee, Lance Titchkosky, "Software cost estimation," 2002. http://www.computing.dcu.ie.

[3] B. W. Boehm, Clark, Horowitz, Brown, Reifer, Chulani, R. Madachy, and B. Steece, Software Cost Estimation with Cocomo II with Cdrom. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1st ed., 2000.

[4] R. Chandrasekaran and R. Kumar, "Article: Application of logistic regression to predict over target baseline of software projects," International Journal of Computer Applications, vol. 44, pp. 1-6, April 2012. Full text available.

[5] R. S. Pressman, Software Engineering: A practitioners Approach. New york: McGrawHill, Inc., 1992.

[6] O. Morgenshtern, T. Raz, and D. Dvir, "Factors affecting duration and effort estimation errors in software development projects", Inf. Softw. Technol., vol. 49, pp. 827-837, Aug. 2007.

[7] R. M. Henry, G. E. McCray, R. L. Purvis, and T. L. Roberts, "Exploiting organizational knowledge in developing is project cost and schedule estimates: An empirical study.," Information & Management, vol. 44, no. 6, pp. 598-612, 2007.

[8] S. Grimstad, M. J rgensen, and K. Mol kken, "Software effortestimation terminology: The tower of babel," Inf. Softw. Technol., vol. 48, pp. 302-310, Apr. 2006.

[9] G. S. M. R. Petr Muslek, Witold Pedrycz, "software cost estimation with granular models," http://www.sigapp.org.

[10] A. Idri, A. Zahi, E. Mendes, and A. Zakrani, "Software process and product measurement," ch. Software Cost Estimation Models Using Radial Basis Function Neural Networks, pp. 21-31, Berlin, Heidelberg: Springer-Verlag, 2008.

[11] Q. Bai, "Analysis of particle swarm optimization algorithm.," Computer and Information Science, vol. 3, no. 1, pp. 180-184, 2010.

[12] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942-1948, 1995.

[13] R. Poli, "Analysis of the publications on the applications of particle swarm optimization," J. Artif. Evol. App., vol. 2008, pp. 4:1-4:10, Jan. 2008.

[14] R. Poli, "Analysis of the publications on the applications of particle swarm optimization," J. Artif. Evol. App., vol. 2008, pp. 3:1-3:10, Jan. 2008.

[15] J. Kennedy and R. C. Eberhart, "New ideas in optimization," ch. The Particle Swarm: Social Adaptation in Information-processing Systems, pp. 379-388, Maidenhead, UK, England: McGraw-Hill Ltd., UK, 1999.

[16] J. Kennedy, "Population structure and particle swarm performance," in In: Proceedings of the Congress on Evolutionary Computation (CEC 2002, pp. 1671-1676, IEEE Press, 2002.

[17] M. Clerc, Standard Particle Swarm Optimization. Open access archive HAL, 2012.

[18] www.cs.ru.ac.za/courses. M. Meissner, M. Schmuker, and G. Schneider, "Optimized particle swarm optimization (opso) and its application to artificial neural network training.," BMC Bioinformatics, vol. 7, p. 125, 2006.

[19] A. A. . H. Guo and H. Liu, "Swarm intelligence: Foundations, perspectives and applications," 2006.

[20] R. M. . J. Kennedy, "Avoiding the pitfalls of local optima: how topologies can save the day."

[21] X. W. . W. W. . X. Zhang and X. Yu, "Annealed particle filter based on particle swarm optimization for articulated three-dimensional human motion tracking,," Optical Engineering, vol. 49, Jan. 2010.

[22] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," Trans. Evol. Comp, vol. 6, pp. 58-73, Feb. 2002.

[23] T. I. Cristian, "The particle swarm optimization algorithm: Convergence analysis and parameter selection," Inf. Process. Lett., vol. 85, pp. 317-325, Mar. 2003.

[24] H. Matsushita and Y. Nishio, "Network-structured particle swarm optimizer considering neighborhood relationships.," in IJCNN, (Department of Electrical and Electronic Engineering, Tokushima University, Tokushima, 7708506, Japan.), pp. 2038-2044, IEEE, 2009.

[25] D. B. Hanchate and Dr. R. S. Bichkar, "Software project scheduling by AGA," International journal of computer Applications, vol. 96, June 2014.

[26] D. B. Hanchate and Dr. R. S. Bichkar, "SPS by combination of crossover types and changeable mutation SGA," International journal of computer Applications, vol. 94, May 2014.

[27] P. H. Dave and H. B. Dave, Design and Analysis of Algorithms. Pearson Education, 2008.

[28] S. S. E. Horowitz, Fundamentals of Computer Algorithms. Galgotia Publications, 1999.

[29] E. A. J. F. Chicano, "Sofware project management with gas," Science Direct, Information Sciences, vol. 177, 2007.

[30] K. N. Vitekar and D. B. Hanchate, "Software project planning using ant colony optimization (spp-aco)," International Journal of Computer Science And Technology, Oct 2013. www.ijcst.com.

[31] F. Van Den Bergh, An Analysis of Particle Swarm Optimizers. PhD thesis, Pretoria, South Africa, South Africa, 2002. AAI0804353.

[32] M. Clerc, Particle Swarm Optimization. ISTE, Wiley, 2013.

[33] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39-43, 1995.

[34] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in Proceedings of the 7th International Conference on Evolutionary Programming VII, EP '98, (London, UK, UK), pp. 591-600, Springer-Verlag, 1998.

[35] M. Pant, R. Thangaraj, and A. Abraha, "Low discrepancy initialized particle swarm optimization for solving constrained optimization problems," Fundamental Inf., vol. 95, pp. 511-531, Dec. 2009.

[36] Y. S. R. Eberhart, "particle swarm optimization: developments, applications and resources,," Proc. IEEE Int. Conf. on Evolutionary Computation, 2001.

[37] Y. Shi and R. C. Eberhart, "A Modified Particle Swarm Optimizer," in Proceedings of IEEE International Conference on Evolutionary Computation, (Washington, DC, USA), pp. 69-73, IEEE Computer Society, May 1998.