ELSEVIER
Available online at www.sciencedirect.com
SciVerse ScienceDirect
Procedia Technology 6 (2012) 231 – 238
Procedia Technology

International Conference on Communication, Computing and Security

An Evolutionary Algorithm for Multi-criteria Resource Constrained Project Scheduling Problem Based On PSO

Tuli Bakshi[a]*, Bijan Sarkar[b], Subir K. Sanyal[b]

[a]Research Scholar, Jadavpur University, Kolkata,west Bengal,India
[b]Professor , Jadavpur University, Kolkata,west Bengal,India

Abstract

This paper is regarding the Particle Swarm Optimization (PSO)-based approach for the solution of the resource-constrained project scheduling problem with the purpose of minimizing cost. In order to evaluate the performance of the PSO based approach for the resource-constrained project scheduling problem, computational analyses are given. As per the results the application of PSO to project scheduling is achievable.

Keywords: Particle Swarm Optimization, Resource-constrained project scheduling, minimizing cost.

1. Introduction

Being a temporary attempt, a project requires to create an unique product, service or result [1]. Temporary emphasizes on definite deadline reaching of which a project may gain its objectives or has already lost its significance of existence. Therefore, a decision maker (DM) has to choose the appropriate alternative among many.

According to Hwang et al.[2], multi-criteria decision making (MCDM)is one of the most widely used decision making methodology.

Many real world projects scheduling problem termed as Resource-Constrained Multi-Project Scheduling Problem (RCMPSP) [3]. To schedule project activities to complete multiple projects in the minimum possible time under presence of resource constraints is the general objective of this type of

* Corresponding author. Tel.: +91-943292656583;
E-mail address: tuli.bakshi@gmail.com

doi:10.1016/j.protcy.2012.10.028

problems. Primarily, mathematical models such as linear programming & dynamic programming have been used to solve & to obtain optimal solution.

It was earlier shown that the scheduling problem subject to resource constraints is NP-Hard [4], refraining exact methods time consuming and inefficient for solving large & real world application type problem.

Hence, meta-heuristic algorithm to generate near optimal solution for large problems has drawn special interest. There are many genetic algorithm (GA) applied for RCMPSP [5, 6, 7, 8] with project duration minimization as objective.

Most recently, the particle swarm optimization (PSO) algorithm has been taken into consideration for solving resource constrained project scheduling problem (RCPSP) and multi-criteria resource constrained project scheduling problem (MRCPSP). PSO, developed by Kennedy and Elbe hart [9] is an evolutionary algorithm which simulates the social behaviour of bird flocking to desired place. It is initialized with a population of random solutions. Each individual particle is tagged with a randomized velocity according to its own and its companions flying experiences. In the PSO, the solution is represented as an optimal solution for the RCMPSP.

Compared with GA, PSO has the following advantages:

1. It has memory that can be retained by all particles in reference to the knowledge of good solution.
2. It has constructive cooperation, between particles, particle in the swarm share information between them.
3. It is easy to implement and quickly converges because of its simplicity.

The current researchers have proposed PSO-based approach to resolve the resource-constrained project scheduling problem. It is an evolutionary algorithm. The results are analyzed and described.

2. Particle Swarm Optimization (PSO)

It is a computational intelligence based optimization technique such as genetic algorithm (GA). It is a population based stochastic optimization technique developed by Kennedy and Eberhart in 1995 [10-13] and inspired by the social behaviour of bird flocking in a group looking for food and fish schooling.

2.1. Some terms related to PSO

The term PARTICLE refers to a member of population which is mass less and volume less m dimensional quantity. It can fly from one position to other in m dimensional search space with a velocity. POPULATION constitutes a number of such particles. The number of iteration for the solution of the problem is same as the number of generations in GA. The fitness function in PSO is same as the objective function for an optimization problem.

In real number space, each individual possible solution can be represented as a particle that moves through the problem space. The position of each particle is determined by the vector x_i and its movement by the velocity of the particle v_i represented in (1) and (2) respectively.

$$X_i^{k+1} = X_i^k + V_i^{k+1} \quad (1)$$

The information available for each individual is based on

1. its own experience (the decisions it has made so far ,stored in memory)
2. the knowledge of performance of other individuals in its neighbourhood.

Since the relative importance of these two information can vary from one decision to other, a random weight is applied to each part and the velocity is determined as in (2)

$$V_i^{k+1} = V_i^k + c1.\text{rand1}.\ (p_{best\ i}{}^k - X_i^k) + c_2.\text{rand2}.\ (g_{best}{}^k - X_i^k) \quad (2)$$

Where, X_i^k = Position vector of a particle i = $[X_{i1}^k, X_{i2}^k, \ldots\ldots X_{im}^k]$ at k^{th} iteration
V_i^K = Velocity vector of a particle i = $[V_{i1}^K, V_{i2}^K, \ldots\ldots V_{im}^K]$ at k^{th} iteration, k = iteration count
$p_{best\,i}^k$ = i th particle has a memory of the best position in the search space at k^{th} iteration . It is computed as $p_{best}\, i^{k+1} = X_i^{k+1}$ if the fitness function of i^{th} at k+1 is less then (for minimum) the fitness function at k^{th} iteration otherwise $p_{bst\,i}^{k+1} = p_{best\,i}^k$ .g_{best}^k = It is that particle which has the minimum value of fitness function (for minimization) among all the particles in k^{th} iteration.
c_1 & c_2 = positive acceleration coefficients more then 1.0.
Normally its value is taken
$c_1 + c_2 = 4$ or $c_1 = c_2 = 2$.
rand1 & rand2 are random numbers between 0.0 & 1.0.
Both the velocity and positions have same units in this case.
The velocity update equation (2) has three components [14]

1. The first component is referred to "Inertia" or "Momentum". It represents the tendency of the particle to continue in the same direction it has been travelling. This component can be scaled by a constant or dynamically in the case of modified PSO.
2. The second component represents local attraction towards the best position of a given particle (whose corresponding fitness value is called the particles best (p_{best}) scaled by a random weight factor c_1.rand1. This component is referred as "Memory" or "Self knowledge".
3. The third component represents attraction towards the position of any particle (whose corresponding fitness value is called global best (g_{best}), scaled by another random weight c_2.rand2. This component is referred to "cooperation" ,"social knowledge", "group knowledge" or "shared information".

The PSO method is explained as above. The implementation of the algorithm is indicated below:

1. Initialize the swarm by assigning a random position to each particle in the problem space as evenly as possible.
2. Evaluate the fitness function of each particle.
3. For each individual particle, compare the particle's fitness value with its p_{best}. If the current value is better than the p_{best} value , then set this value as the p_{best} and the current particle's position X_i as p_{best} i.
4. Identify the particle that has the best fitness value and corresponding position of the particle as g_{best}.
5. Update the velocity and positions of all the particles using equations (1) & (2).
6. Repeat steps i) to v) until a stopping criterion is met (e.g. maximum number of iterations or a sufficient good fitness value).

On implementation of PSO following considerations must be taken into account to facilitate the convergence and prevent an "explosion" (failure) of the swarm resulting in the variants of PSO.

2.2. Selection of Maximum velocity:

At each iteration step, the algorithm proceeds by adjusting the distance (velocity) that each particle moves in every dimension of problem space. The velocity of a particle is a stochastic variable and it may create an uncontrolled trajectory leading to "explosion". In order to damp these oscillations upper and lower limits of the velocity V_i is defined as
if $V_{id} > V_{max}$ then $V_{id} = V_{max}$
else if $V_{id} < -V_{max}$ then $V_{id} = -V_{max}$
Most of the time, the value V_{max} is selected empirically depending on the characteristic of the problem. It is important it note that if the value of this parameter is too high, then the particle may move erratically, going beyond a good solution , on the other hand, if V_{max} is too small, then the particle movement is

limited and it may not reach to optimal solution. The dynamically changing V_{max} can improve the performance given by

$V_{max} = (X_{max} – X_{min})/N$

Where X_{max} and X_{min} are maximum and minimum values of the found so far and N is the number of intervals.

2.3. Selection of Acceleration Constants:

c_1 & c_2 are the acceleration constants; they control the movement of each particle towards its individual and global best positions. Small values limit the movement of the particles, while larger values may cause the particle to diverge. Normally the constants $c_1 + c_2$ limited to 4. If it is taken more than 4 the trajectory may diverge leading to "Explosion". In general a good start is when $c_1 = c_2 = 2$.

2.4. Selection of Constriction Factor or Inertia Constant

Experimental study performed on PSO shows that even the maximum velocity and acceleration constants are correctly chosen, the particles trajectory may diverge leading to infinity, a phenomenon known as "Explosion" of the swarm. Two methods are to control this explosion (a) Inertia control and (b) Constriction factor control, the two variants of PSO.

a. Inertia Constant

The velocity improvement represented by equation (2) is modified [15-17] and written as

$$V_i^{k+1} = W.V_i^k + c_1.\text{rand1}.\ (p_{best\ i}^{\ k} – X_i^k) + c_2.\text{rand2}.(g_{best}^{\ k} – Xi^k) \quad (3)$$

The first right hand side part (velocity of previous iteration) of equation (3) multiplied by a factor W is known as "Inertia Constant". It can be fixed or dynamically changing. It controls the "Explosion" of search space. Initially it is taken as high value (0.9) which finds the global neighborhood fast. Once it is found that it is decreasing gradually to o.4 in order to find narrow search as shown in equation (4)

$$W = wmax – (wmax – wmin)*itr/itrmax \quad (4)$$

Where wmax = 0.9,wmin = 0.4,itrmax= maximum iterations, itr = current iteration

Since the weighting factor W is changing iteration wise it may be called as Dynamic PSO.

(b) Constriction Factor

This is another method of control of "Explosion" of the swarm. The velocity in equation (2) is redefined using constriction factor developed by Clark and Kennedy [17], is represented in equation (5) as

$$V_i^{k+1} = K*(V_i^k + c_1.\text{rand1}.(p_{best\ i}^{\ k} – X_i) + c_2.\text{rand2}.(g_{best}^{\ k} – X_i)) \quad (5)$$

Where K is known as constriction factor

$$K = 2/(abs(2 – c – sqrt(c^2 – 4*c)) \quad (6)$$

Where, $c = c_1 + c_2 > 4.0$

Typically when this method is used, c is set to 4.1 and value of K comes out to be 0.729, In general, the constriction factor improves the convergence of the particle by damping oscillations. The main disadvantage is that the particles may follow wider cycles when $p_{best\ I}$ is far from g_{best} (two different regions).A survey is given in reference [18].The present problem is discussed in next section.

3. Problem formulation

Resource optimization problems in project management have been solved either as resource levelling or as resource allocation problems. Classical resource-constrained project scheduling problem can be described as follows: Given are M projects / jobs in a process and N_m activities in the project m= 1 …M. There are T times required to complete the jobs. Each activity must be processed for t_j time unit, where pre-emption is not allowed. During this time period a constant amount of r_{mk} unit of time t is occupied. All the values are supposed to non-negative integers. The objective is to minimize the total cost. The model is formulated as follows:

$$\min C_{\max}$$

$$\text{s.t } \forall i \in N_{m,} t_i \geq o$$

$$\forall i \in N_{m,} t_i + r_{mk} \leq C_{\max}$$

where $C_{\max}$ denote the total cost which has to be minimized and all $t_i >= 0$

4. Case Study Analysis & Problem definition

In a process there are three jobs to be carried out. It is assumed that each job is completed in a fixed time. Say for example:

Job A takes 20 days (T_A)
Job B takes 30 days (T_B)
Job C takes 50 days (T_C)

It is further assumed that Job A & Job B or Job A, Job B & Job C or Job B & Job C can work at a time in addition to they can work individually. Whereas Job A & Job C cannot work together and Job B cannot work alone.

The cost involved in carrying out the job is indicated as below:

Job A = 20
Job A & Job B = 25
Job A & Job B & Job C = 50
Job B & Job C = 30
Job C = 20

The jobs are carried out as shown in Fig 1. That is Only Job A for t_1 time.

Job A & Job B for t_2 time
Job A & Job B & Job C for t_3 time.
Job B & Job C for t_4 time.
Job C for t_5 time.

No time is negative. That is $t \geq 0$. The equality constraints are:

$$t_1 + t_2 + t_3 = 20$$
$$t_2 + t_3 + t_4 = 30$$
$$t_4 + t_5 = 50$$

It is to find the project schedule for completing the project such that the cost involved in project completion is minimum, subject to the above mentioned equality and inequality constraints.

5. Result set of Case Study

Here we are taking the following as input and getting the final cost:

$$\text{Initial value of } P_{best} = \begin{matrix} 8,6,6,18,32 \\ 5,10,5,15,35 \\ 15,2,3,25,25 \\ 5,5,10,15,35 \\ 10,5,5,20,30 \\ 12,3,5,22,28 \end{matrix}$$

V_{max} = 10
V_{min} = -10
w_{max} = 0.9000
w_{min} = 0.4000
itr_{max} = 10

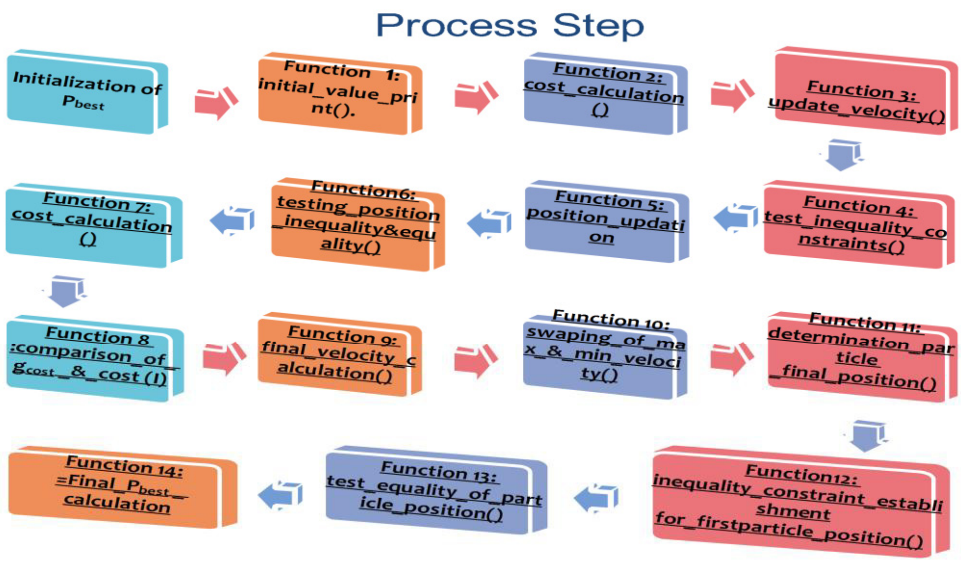
$$\text{Inequality tested position} = \begin{matrix} 30,0,0,40,10 \\ 30,0,0,40,10 \\ 30,0,0,40,10 \\ 30,0,0,40,10 \\ 30,0,0,40,10 \\ 30,0,0,40,10 \end{matrix}$$

Equality condition tested

$$\text{result} = \begin{matrix} 20,0,0,30,20 \\ 20,0,0,30,20 \\ 20,0,0,30,20 \\ 20,0,0,30,20 \\ 20,0,0,30,20 \\ 20,0,0,30,20 \end{matrix}$$

Final minimum cost = **1500 1500 1500 1500 1500.**

6. Solution Process Steps of Case Study Problem



7. Conclusion

This paper presented a new approach for cost minimization problem. The conventional method such as CPM , PERT and others are mainly used for minimizing the duration of project to solve unconstrained project scheduling problem. But it is difficult to use these methods for solving more general scheduling problems mainly cost optimization problem. In this paper the current researchers presented that the application of PSO is possible in case of many general problems related to project scheduling without much obstacles.PSO is meta-heuristic approach. So it can be concluded that this meta-heuristic approaches are successful to solve project scheduling problems and the inclusion of much problem specific knowledge is needed for the heuristic. Future research may be included to the development of meta-heuristic algorithms for the RCMPSP and their comparative study with the PSO approach.

References

[1]. Bakshi T.,Sarkar B., MCA Based Performance Evaluation of project selection, International Journal of software engineering & Applications (IJSEA), Vol.2, No2,2011,pp-14-22.
[2]. C.L. Hwang & K.P.Yoon, Multiple Attribute Decision Making and Introduction, London, Sage publication,1995,pp2.
[3]. Deng Lin-yi, Wang Yun –long, Lin Yan, A Particle Swarm Optimization Based on Priority Rule for Resource-Constrained Multi-Project Scheduling Problem,978-1-4244-1734-6/08/$25.00@2008 IEEE,pp-1038-1041.
[4]. M. R. Garey, D. S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, New York, 1979.
[5]. W. H. Ip, Y Li, K. F. Man, K.S. Tang, Multi-product planning and scheduling using genetic algorithm approach, Computer & Industrial Engineering, Vol.38, No.2, 283-296, 2000.
[6]. P. Pongcharoen, C Hicks, P M Braiden, The development of genetic algorithm for the capacity scheduling of complex product, with multiple levels of product structure, European Journal of Operational Research, Vol.152, No.l, 215-225, 2004.
[7]. F. S. C. Lam, B. C. Lin, C. Sriskandarajah, H.Yan, Scheduling to minimize project design time using a genetic algorithm, International Journal of Production Research, Vol.37, No.6, 1369-1386, 1999.
[8]. M. Zhuang, A. Yassine, Task scheduling of parallel development projects using genetic algorithms, American Society of Mechanical Engineers Design Automation Conference. Salt Lake City, 1-11, 2004.
[9]. Kennedy J, Eberhart R C, A discrete Binary Version of the Particle Swarm Algorithm,In Proc.1997 Conf. On System, Man and Cybernetics Piscataway, NJ:IEEE Service Center, 1997,4104-4109.
[10]. Y.Shi and R.C. Eberhart," Particle Swarm Optimization: Developments, Applications And Resources", Proceedings of the 2001 Congress on Evolutionary Computation,Vol. 1, pp. 81-86, 2001.
[11]. R. C. Eberhart and Y. Shi, "Comparing Inertia Weights and Constriction Factor in Particle Swarm Optimization ", Proceedings of the 2000 Congress on Evolutionary Computation, Vol. 1, pp. 84-88, 2000.
[12]. J. Kennedy and R. Eberhart, "Particle Swarm Optimization ", Proc. Int. Conf. Neural Networks (ICNN), Nov. 1995, Vol. 4, pp. 1942-1948.
[13]. R, Eberhart and J. Kennedy, "A New Optimizer Using Particle Swarm Theory", Proc. 6th Int. Symp. Micro Machine and Human Science (MHS), Oct. 1995, pp.39-43.
[14]. D. Boiringer and D. Werner, "Particle Swarm Optimization versus Genetic Algorithms for Phase Array Synthesis", IEEE Trans. Antennas Propagat. Vol. 52, No. 3, pp. 771-779, Mar. 2004.
[15]. Y. Shi and R. Eberhart, "A Modified Particle Swarm Optimization", Proc. IEEE World Cong. Comput. Intell., May 1998, pp. 69-73.
[16]. Y. Shi and R. Eberhart, "Empirical Study of Particle Swarm Optimization", Proc. IEEE Cong. Evol. Comput. July 1999, Vol. 3, pp. 1945-1950.
[17]. M. Clerc and J. Kennedy, "The Particle Swarm Explosion, Stability and Convergence in a multidimensional Complex Space", IEEE Trans. Evol. Comput. Vol. 6, No. 1, pp. 58-73, Feb. 2002.
[18]. Yamille del Valle et.al. "Particle Swarm Optimization : Basic Concepts, Variants And Applications in Power Systems", IEEE Trans. on Evolutionary Computation,Vol. 12, No. 2, April 2008.

Appendix A. Solution Technique of the Case Study

```
Initialization of
Pbest............................................
for i:=1 to n do
for j:= 1 to m do
        update Pbest by initial value of
particle
done
done
Function 1: initial value print().
```

```
{
for i:=1 to n do
print the values
}
Initialization cost of ......................................
Function 2: cost calculation()
{
for itr :=1 to 10 do
        for i:= 1 to n do
                for j:=1 to m do
                done
        done
done
}
print the values of gbest , gcost

Function 3: update velocity()
{
set v:=velocity() for n particle
}
Function 4: test inequality constraints
{
set v:=testval () for n particles
}
Function 5: position updation
{
set T:=position()for n particles
}
Function6:
testing position inequality&equality()
{  set  T:=test  inequality  position()  for  n
inequalities
   set  T:=  test  equality  position()  for  n
equalities }
for i:=1 to n do
assign Pcost  with cost (I)
done
Function 7: cost calculation()
{
        set cost:=initial cost
}
for i:=1 to n do
set cost I = 0
fot j:=1 to m do
set cost (I) = old (cost I)+a(J) *T(I,J)
done
done
Function 8 :comparison of  gcost  & cost (I)
{       Initialize gcost
for i:=1 to n-1do
if gcost > cost (I+1)
then update gcost by cost (I+1)
done
}
Function 9: final velocity calculation()
{       Define constraints
for i:=1 to n do
for j:=1 to m do
determine the velocity by equation (2)
        done
```

```
done
}
Function 10: swaping of max & min velocity()
{       for i:=1 to n do
for j:=1 to m do
compare and update vmax
compare and update vmin
                done
done
}
Function      11:      determination particle
 final position()
{       for i:= 1 to n do
                for j:= 1 to m do
update position of particle
                done
done
}
Function12:
inequality constraint establishment
for firstparticle position()
{       for i:=1 to n do
checking whether the particle position within
limit
done
for i:=1 to n do
for j:=1 to m do
checking whether the particle is in negative
position
then if yes
then make it zero
                 done
done
}
Function                                  13:
test equality of particle position()
{       set constraints
for i:= 1 to n do

check the constraint and update for row1
check the constraint and update for row2
check the constraint and update for row3
check the constraint and update for row4
check the constraint and update for row5
check the constraint and update for row6
done
}
Function 14:=Final Pbest   calculation
{for i:=1 to n do
if Pcost(I) > cost(I) { do
for j:=1 to m
set Pbest = position of the particle
done}
else
{       for j:=1 to m do
set Pbest(I,J) = TT(I,J)
done
}
done
}
```