Solving Resource-Constrained Project Scheduling Problem with Particle Swarm Optimization

Sylverin Kemmoé Tchomté, Michel Gourgand, Alain Quilliot
LIMOS, Campus Scientifique des Cézeaux, France, {kemmoe, gourgand, quilliot}@isima.fr

This paper presents a Particle Swarm Optimization (PSO) algorithm for solving Resource-Constrained Project Scheduling Problems (RCPSP). The PSO model is a new population based optimization strategy introduced by Kennedy and Eberhart in 1995. The PSO is a cooperative and competitive algorithm who belongs to the class of the evolutionary algorithms. We here specialize the algorithm of PSO to the particular case of the RCPSP. We have redefined the PSO operators (the addition of two velocities, the difference between two positions, the external multiplication between a scalar and a velocity). We propose in this paper, an extension of the PSO system that integrates a new displacement of the particles and we highlight a relation between the coefficients for each dimension between the classical PSO algorithm and the extension. The experiments on instances from the PSPLIB show that the proposed PSO algorithm is able to solve to optimality most problems of the series and close to the best known solutions on the hardest instances. This seems to indicate that this general algorithm is a good candidate for solving scheduling problems.

Keywords: Particle Swarm Optimization, RCPSP, Evolutionary Algorithms.

1 Introduction

In this paper, we present an algorithm called Particle Swarm Optimization (PSO) for solving the Resource-Constrained Project Scheduling Problem (RCPSP). This problem is interesting for two reasons : on the one hand, it is the core problem of many real-life industrial scheduling applications, where resources can handle several tasks at a time. On the other hand, it is an academic problem for which there exists a variety of algorithms and benchmarks, and new algorithms can be tested against a rigorous gauge. Hence, in order to evaluate a scheduling algorithm designed for particular industrial purposes, if the RCPSP can be recognized in a sub-case of the industrial problem definition, it is interesting measure how well the general algorithm solves the particular RCPSP benchmarks. The PSO algorithm is a stochastic population based optimization approach, first published by Kennedy and Eberhart in 1995. They simulated the scenario in which birds search for food and observed their social behavior. They perceived that in order to find food the individual members determined their velocities by two factors, their own best previous experience and the best experience of all other members. This is similar to the human behavior in making decision where people consider their own best past experience and the best experience of how the other people around them have performed. According to the above concept, Kennedy and Eberhart developed the so-called PSO for optimization of continuous nonlinear functions in 1995.

2 Resource-Constrained Project Scheduling Problem

A Resource-Constrained Project Scheduling Problem (RCPSP) is defined by a set or tasks T and a set of resources R. Each task i is performed on some resources k (possibly many) of which it requires a given amount r_{ik}. Each resource k is available in a given quantity R_k. For each task

i, a duration d_i is given as well as a set of precedence constraints ($i << j$). constraining i to be finished before j starts. All data are integer. The problem is solved when one has found a set of integer starting dates S_i such that :

$$for \quad all \quad precedence \quad constraints \quad with \quad i << j : \quad S_i + d_i \leq S_j \tag{1}$$

$$for \quad all \quad resources \quad k, \quad and \quad all \quad time \quad t : \quad \sum_{\{i|S_i \leq t \leq S_i + d_i\}} r_{ik} \leq R_k \tag{2}$$

The purpose is to minimize the latest end time ($S_i + d_i$) over all tasks i (which is called the makespan of the schedule). A RCPSP problem is known to be NP hard [5] and NP-Complete [6]. As State-of-the-art of the RCPSP, a review of the heuristics can be found in [1]. Notation, classification, models, and methods for the RCPSP are given in [2] and [3].

3 Particle Swarm Optimization algorithm

Particle Swarm Optimization (PSO) is a recently proposed algorithm by J. Kennedy and R. Eberhart in 1995, motivated by social behavior of organisms such as bird flocking and fish schooling. PSO algorithm is not only a tool for optimization, but also a tool for representing socio-cognition of human and artificial agents, based on principles of social behavior. Some scientists suggest that knowledge is optimized by social interaction and thinking is not only private but also interpersonal. PSO as an optimization tool, provides a population-based search procedure in which individuals called particles change their position (state) with time. In a PSO system, particles fly in a multidimensional search space. During flight, each particle adjusts its position according to its own experience, and according to the experience of K neighbors, making use of the best position encountered by itself and its neighbors. The basic PSO model consists of a swarm of particles moving in an n-dimensional, real valued search space of possible problem solutions. For the search space, in general, a certain quality measure, the fitness, is defined making it possible for particles to compare different problem solutions. Every particle i at the time t has the following characteristics: $X_{i,t}$ is the position vector; $V_{i,t}$ is the velocity vector; $P_{i,t}$ is the small memory storing its own best position seen so far; $G_{i,t}$ is the global best position obtained through communication with its fellow neighbor particles. This information flow is obtained by defining a neighborhood topology on the swarm.
The intuition behind the PSO model is that by letting information about good solutions spread out through the swarm, the particles will tend to move to good areas in the search space. At each time step t the velocity is updated and the particle is moved to a new position. This new position is simply calculated as the sum of the previous position and the new velocity:

$$X_{i,t+1} = X_{i,t} + V_{i,t+1} \tag{3}$$

The update of the velocity from the previous velocity to the new velocity is determined by:

$$V_{i,t+1} = \underbrace{c_1 * V_{i,t}}_{momentum} + \underbrace{c_2 * r_2 * (P_{i,t} - X_{i,t})}_{local \quad information} + \underbrace{c_3 * r_3 * (G_{i,t} - X_{i,t})}_{global \quad information} \tag{4}$$

where the parameter c_1 introduced by Shi and Eberhart [7] is the inertia weight, it controls the magnitude of the old velocity $V_{i,t}$. The parameters c_2 and c_3 are real numbers chosen uniformly and at random in a given interval, usually $[0; 1]$. These values determine the significance of $P_{i,t}$ and $G_{i,t}$ respectively. The three velocities components are defined as follows: $c_1 * V_{i,t}$ serves as a momentum term to prevent excessive oscillations in search direction; $c_2 * r_2 * (P_{i,t} - X_{i,t})$ refers to as the cognitive

component, it represents the natural tendency of individuals to return to environments where they experienced their best performance; $c_3 * r_3 * (G_{i,t} - X_{i,t})$ refers to as the social component, it represents the tendency of individuals to follow the success of other individuals.
For the neighborhood system, there are many ways to define it. Kennedy,[8] distinguish two classes: *Physical* neighborhood, which takes distances into account. In practice, distances are recomputed at each time step, which is quite costly, but some clustering techniques need this information. *Social* neighborhood, which just takes *relationships* into account. In practice, for each particle, its neighborhood is defined as a list of particles at the very beginning, and does not change. Note that, when the process converges, a social neighborhood becomes a physical one.

4 Method proposal

We propose a resolution method of the RCPSP problem based on PSO algorithm. This method is an extension of PSO algorithm in which particles have a new displacement technique.

4.1 Proposal of a New Displacement of the particles

The intuition behind this new particle displacement is that by letting information about two good solutions $(P_{i,t}, G_{i,t})$ spread out through the swarm, the particles will tend to move around these solutions in the search space. Let us note by $S_{i,t}$ and $T_{i,t}$ the intermediate positions of particle i at the step t from the position $X_{i,t}$ to the position $X_{i,t+1}$. The figure 1 illustrates of part A this new displacement and the classical one on part B (each co-ordinate of V_{it}, $(P_{it} - X_{it})$ and $(G_{it} - X_{it})$ has its own coefficient).

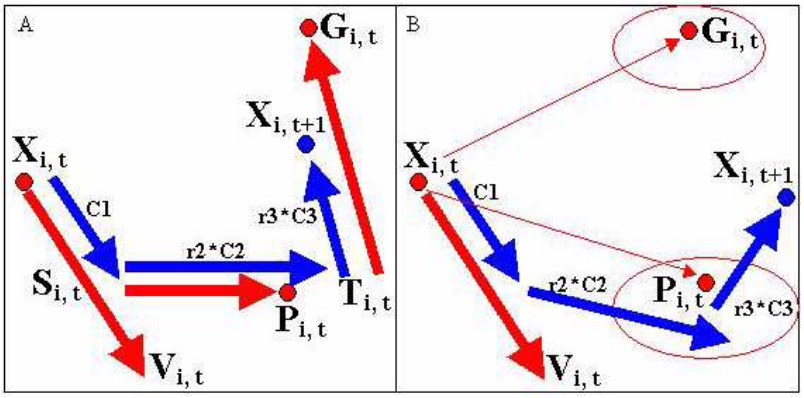

Figure 1: Particle displacement in the swarm space

At each time step t the position is updated, This new position is simply calculated with the intermediate positions, we obtain: $S_{i,t} = X_{i,t} + c_1 * V_{i,t}$; $T_{i,t} = S_{i,t} + c_2 * r_2 * (P_{i,t} - S_{i,t})$; $X_{i,t+1} = T_{i,t} + c_3 * r_3 * (G_{i,t} - T_{i,t}) \implies$

$$V_{i,t+1} = \underbrace{c_1 * (1 - c_2 * r_2) * (1 - c_3 * r_3)}_{\alpha} * V_{i,t} + r_2 * \underbrace{c_2 * (1 - c_3 * r_3)}_{\beta} * (P_{i,t} - X_{i,t}) + r_3 * \underbrace{c_3}_{\gamma} * (G_{i,t} - X_{i,t})$$

$$\text{In short} \quad V_{i,t+1} = \alpha * V_{i,t} + \beta * r_2 * (P_{i,t} - X_{i,t}) + \gamma * r_3 * (G_{i,t} - X_{i,t}) \tag{5}$$

$$\text{with} \begin{cases} \alpha & = c_1 * (1 - c_2 * r_2) * (1 - c_3 * r_3) \\ \beta & = c_2 * (1 - c_3 * r_3) \\ \gamma & = c_3 \end{cases} \quad (6)$$

4.2 Redefinition of PSO operators

The PSO algorithm has been invented to solve continuous non linear functions. Clerc [9] has proposed to extend some operators to solve combinatorial optimization problem (in particular the Traveling Salesman Problem - TSP). For this problem in particular, an exchange of two cities to visit give always a feasible solution but it is not the case of the RCPSP, that contains precedence relations between activities. This is why, while basing us on work of Clerc, we redefine all the operators of the PSO algorithm to treat the case of combinatorial problems with precedence constraints.

- **Position:** A position X is a sequence of activities checking the constraints of precedence ($X = (x_1, ..., x_n)$).
- **Velocity:** A velocity V is a list of couples of activities or transpositions ($V = ((x_i, x_j)_{i,j \in \{1,...,n\}})$). (x_i, x_j) means that exchange activities at the positions x_i^{th} and x_j^{th}. it is possible to have an empty list. The number of transpositions is equal to the norm of the velocity.
- **Opposite of a velocity:** The opposite of a velocity is a velocity. It means to take the same transpositions as in V, but in reverse order. It is easy to verify that the opposite of the opposite velocity give the same velocity.
- **Move of a particle:** the move of a particle is obtained by the application of a velocity to position. The result is another position. It is found by applying the first transposition of V to X, then the second one to the result etc. For example : we consider a position $X = (1, 5, 2, 4, 3)$ with the precedence relation $2 \prec 3$. We consider also a velocity with one transposition (2;5), the activities at the positions 2 and 5 are respectively 5 and 3. Exchange the activities 5 and 3 will violate the relation of precedence between the activities 2 and 3. we will right shift the activity 5 as long as the precedence relation is not violated and we will left shift the activity 3 as long as the precedence relation is not violated. We obtain the position result $X' = (1, 2, 3, 4, 5)$.
- **Difference between two positions:** The difference between two positions is defined as the velocity. To obtain this velocity it is necessary to build an algorithm. We propose a non optimal algorithm that gives this velocity: $NORM$ gives the norm of the velocity obtained, $Exchange()$ is a procedure that allows to permute two activities.
- **Sum of two velocities:** The sum of two velocities V1 and V2 gives a velocity. The list of transpositions which contains first the ones of V1, followed by the ones of V2.
- **External multiplication of a velocity by a scalar:** The external multiplication of a scalar c by velocity V is a velocity V'. This velocity is obtained as follows : If $c = 0$ then $V' = c * V = \emptyset$. If $0 < c < 1$ then the velocity V' is obtained by *truncating* the velocity V and the norm is equal to the integer part of $c * ||V||$. If $c = c' + r > 1$ with c' is an integer number and $0 < r < 1$ a fractional number, we duplicate the velocity V for c' times to obtain V1 and we truncate V according to the number r to obtain V2. the velocity result is the sum of V1 and V2.

Algorithm 1 Difference between two positions X and Y (V = Y - X)

DEBUT
$NORM := 0$
for $j = 1, n$ **do**
  if $X(j) \neq Y(j)$ **then**
    $k := j + 1, trouve := 0$
    while $k \leq n$ AND $trouve = 0$ **do**
      if $X(k) \neq Y(j)$ **then**
        $trouve := 1$
      else
        $k := k + 1$
      end if
    end while
    $NORM := NORM + 1, V \longleftarrow (j, k), Exchange(j, k)$
  end if
end for
END

4.3 Sequence Evaluation

The PSO algorithm allows to update the positions of each particle of the swarm, each position has to be evaluated and a criterion has to be assigned. There is many methods to evaluate a sequence of a combinatorial problem. We propose to use an evaluation algorithm based on non strict order:

Algorithm 2 Sequence evaluation

BEGIN
$ED(job) := 1 \quad \forall job$
for $j = 1, n$ **do**
  $job := X(j), t := ED(job), trouve := 0$
  while $trouve = 0$ **do**
    $start := t, end := t + PT(job) - 1$
    if $ResourceAvailable(job, start, end)$ **then**
      $trouve := 1$
    else
      $t := t + 1$
    end if
  end while
  $ST(job) := deb - 1, ET(job) := ST(job) + PT(job), ResCons(job)$
  for $\forall jobsuc \in Succ(job)$ **do**
    $ED(jobsuc) := ET(job)$
  end for
end for
END

$ED(job)$ is a vector that gives for an activity called job the earliest date of its beginning; $PT(job)$ is a vector that gives for an activity called job the processing time; $ST(job)$ is a vector that gives for an activity called job the start time; $ET(job)$ is a vector that gives for an activity called job the end time; $ResCons(job)$ is a procedure that allows the resource consumption by the

activity called job; $Succ(job)$ is a procedure that gives all the immediate successors of the job.

5 Computational Analysis

This section reports on a computational comparison of several of the heuristics. For test instances, we have used the standard sets j30 and j60 provided on the website (http://129.187.106.231/psplib). The projects in these test sets consist of 30 and 60 activities, respectively and consist of 480 instances each (10 replications). The number of generated and evaluated schedules are fixed to 1000 and 5000, respectively. Tables 1 and 2 present a comparison of PSO to state-ofthe-art heuristic algorithms. We have looked at all the algorithms referred to in [11]. The coefficients ($\alpha = 0.047, \beta = 0.378, \gamma = 1.494$) used in this paper are obtained by applying equation (6) on those ($c_1 = 0.729, c_2 = 1.494, c_3 = 1.494$) provided by [21]. According to [9], the best swarm and neighbourhood sizes are depending on the number of local minimums of the objective function. Usually, we simply do not have this information, except the fact that there are at most N-1 such minimums. In this study, we fix the swarm size at N particles where N the number of dimension of the search space. Table 1 summarises the average deviations from optimal makespan in j30. The results indicate that PSO is capable of providing near optimal solutions. The average CPU time for PSO is 0.2 and 2.18 seconds for 1000 and 5000 evaluations respectively, what are very competitive.

Algorithm type	Authors	Iterations 1000	Iterations 5000
PSO	**Kemmoe et al.**	**0.26**	**0.21**
GA, TS, path reli.,[10]	Kochetov and Stolyar (2003)	0.1	0.04
Hybrid GA, [11]	Valls et al. (2007)	0.27	0.06
GA, [12]	Alcaraz and Maroto (2001)	0.33	0.12
DJGA, [13]	Valls et al. (2005)	0.34	0.20
Sampling + BF/FB, [14]	Tormos and Lova (2003a)	0.25	0.13
Tabu Search, [16]	Nonobe and Ibaraki (2002)	0.46	0.16
Sampling + BF, [17]	Tormos and Lova (2001)	0.30	0.16
Self-adapting GA, [19]	Hartmann (2002)	0.38	0.22
Activity list GA, [18]	Hartmann (1998)	0.54	0.25
Sampling + BF, [15]	Tormos and Lova (2003b)	0.3	0.17

Table 1: Average deviations from optimal solution - J=30

Table 2 summarises the average deviations from critical path based lower bound in j60. The critical path makespan lower bound is obtained by computing the length of a critical path in the resource relaxation of the problem. The results indicate that our algorithm outperforms all algorithms in block one for all schedule limits. PSO is capable of providing near optimal solutions. The average CPU time for PSO is 1.77 and 6.18 seconds for 1000 and 5000 evaluations respectively.

Algorithm type	Authors	Iterations 1000	Iterations 5000
PSO	**Kemmoe et al.**	**9.52**	**9.01**
Hybrid GA	Valls et al.	11.56	11.10
GA, TS, path reli.,[10]	Kochetov and Stolyar (2003)	11.71 11.17	
DJGA, [13]	Valls et al. (2005)	12.21	11.27
Self-adapting GA, [19]	Hartmann (2002)	12.21	11.70

Activity list GA, [18]	Hartmann (1998)	12.68	11.89
Sampling + BF/FB, [14]	Tormos and Lova (2003a)	11.88	11.62
Sampling + BF, [15]	Tormos and Lova (2003b)	12.14	11.82
GA, [12]	Alcaraz and Maroto (2001)	12.57	11.86
Sampling + BF, [17]	Tormos and Lova (2001)	12.18	11.87
SA, [20]	Bouleimen and Lecocq (2003)	12.75	11.90

Table 2: Average deviations from critical path - J=60

6 Conclusion

We have proposed in this paper, a new particle displacement for PSO. We suggest that the proposal extension of PSO balance better the process of exploitation (intensification) and exploration (diversification). We have developed a mathematical formulation which preserves the methodical mind of PSO. This formulation provides a correspondence between the coefficients of our proposal method and those of the classical PSO.
We have applied this extension of PSO to the combinatorial optimization problem of RCPSP. These problems contain constraints of precedence and the general framework provided by clerc [9] can not be applied. We propose in this paper to adapt this framework to hold prcedence constraints by redefining the operators to obtain only feasible sequences.
The computational results show that PSO is a fast and high quality algorithm. Our algorithm is competitive with other state-of-the-art heuristics for the instance set j30. For the instance set j60, our algorithm outperforms all state-of-the-art algorithms for the RCPSP.
The results quality can still be increased if we add the two mechanisms regularly present in the published metaheuristics : an initial population (swarm) built by an heuristic and a local search method exploiting the specific knowledge of the problem (critical path method for example).
The future research of this paper is to provide more computational analysis of this extension on j90 and j120 sets of RCPSP and on others combinatorial optimization problems (Flow-Shop, Job-Shop, etc).

References

[1] S. Hartmann, and R. Kolisch (1998), Experimental Evaluation of State-of-the-art Heuristics of Resource-Constrained Scheduling Problem, *Invited paper of the INFORMS spring conference - Cincinatti may 2-5*, N 476.

[2] P. Brucker, and A. Drexl, and R. Mohring, and K. Neumann, E. Pesch (1999), Resource-constrained project scheduling: Notation, classification, models, and methods, *European Journal of Operational Research* **112**, 3 – 41.

[3] R. Kolisch, and R. Padman (2001), An integrated survey of deterministic project scheduling, *Omega* **29**(3), 249 – 272.

[4] P. Brucker (1999), An integrated survey of deterministic project scheduling, *Omega* **29**(3), 249 – 272.

[5] J. Blazewicz, J.K Lentra, et A.H.G Rinnoy Kan (1983), Scheduling projects subject to resource constraintes: classification and complexity, *Discrete Applied Mathematics* **5**, 11 – 24.

[6] P.Brucker, S. Knust, D. Roper, and Y. Zinder (1999), Scheduling UET task system with concurrency on two parallel identical processors, *Mathematical Methods of Operation Research*.

[7] Y. Shi and R. Eberhart (1998), Parameter Selection in Particle Swarm Optimization. *Annual conference Evolutionary programing*, San Diego.

[8] J. Kennedy (1999), Small worlds and megaminds: Effects of neighborhood topology on particle swarm performance, *Proceedings of the IEEE Congress on Evolutionary Computation*, 1931 – 1938.

[9] M. Clerc (2004), Discrete Particle Swarm Optimization, In G.C. Onwubolu and B.V. Babu, editors, *New Optimization Techniques in Engineering*, Springer-Verlag, 219 – 204.

[10] A. Kochetov and Y. Stolyar (2003), Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem, In: *Proceeding of Workshop on Computer Science and Information Technologies*.

[11] V. Valls and F. Ballestin and S. Quintanilla (2007), A hybrid genetic algorithm for the resource-constrained scheduling problem, *European Journal of Operational Research*, doi:10.1016/j.ejor.2006.12.033

[12] J. Alcaraz and C. Maroto (2001), A robust genetic algorithm for resource allocation in project scheduling, *Annals of Operations Research* **102**, 83 - 109.

[13] V. Valls and F. Ballestin and S. Quintanilla (2005), *Justification and RCPSP: A technique that pays*, *European Journal of Operational Research* **165**, 375 - 386.

[14] P. Tormos and A. Lova (2003a), Integrating heuristics for resource constrained project scheduling: One step forward, Technical report, Department of Statistics and Operations Research, Universidad Politekcnica de Valencia.

[15] P. Tormos and A. Lova (2003b), An efficient multi-pass heuristic for project scheduling with constrained resources, *International Journal of Production Research* **41**, 1071 - 1086.

[16] K. Nonobe and T. Ibaraki (2002), Formulation and tabu search algorithm for the resource constrained project scheduling problem (RCPSP). In: Ribeiro, C.C., Hansen, P. (Eds.), *Essays and Surveys in Metaheuristics*, Kluwer Academic Publishers, 557 - 588.

[17] P. Tormos and A. Lova (2001), A competitive heuristic solution technique for resource-constrained project scheduling, Annals of Operations Research **102**, 65 - 81.

[18] S. Hartmann (1998), A competitive genetic algorithm for resource-constrained project scheduling, Naval Research Logistics **45**, 733 - 750.

[19] S. Hartmann (2002), A self-adapting genetic algorithm for project scheduling under resource constraints, *Naval Research Logistics* **49**, 433 - 448.

[20] K. Bouleimen and H. Lecocq (2003), A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version, *European Journal of Operational Research* **149**(2), 268 - 281.

[21] M. Clerc (1999), The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. Congress on Evolutionary Computation, Washington, DC, Piscataway, NJ: IEEE Service Center., 1951 - 1957.