Scientific Journal of Impact Factor(SJIF): 3.134

e-ISSN(O): 2348-4470
p-ISSN(P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 2, Issue 5, May -2015

A Literature Review on Solving Job Shop Scheduling Problem with Particle Swarm Optimization (PSO)

Abhijeet Thakur[1] & Dr. V N Bartaria[2],

[1]*P.G. Scholar,* [2]*Professor & HOD, Department of Mechanical Engineering, Lakshmi Narain College of Technology, Bhopal, India*

Abstract :*Most previous research into the job-shop scheduling problem has concentrated on finding a single optimal solution (e.g., makespan), even though the actual requirement of most production systems requires multi-objective optimization. The aim of this paper is to construct a particle swarm optimization (PSO) for an elaborate multi-objective job-shop scheduling problem. The original PSO was used to solve continuous optimization problems. Due to the discrete solution spaces of scheduling optimization problems, the authors modified the particle position representation, particle movement, and particle velocity in this study. The modified PSO was used to solve various benchmark problems. Test results demonstrated that the modified PSO performed better in search quality and efficiency than traditional evolutionary heuristics.*

Keywords : *Job-shop scheduling, Particle Swarm Optimization, Multiple Objectives*

1 INTRODUCTION

1.1 Job-shop scheduling problem

The single-objective JSP has attracted wide research attention. Most studies of single-objective JSPs result in a schedule to minimize the time required to complete all jobs, i.e., to minimize the makespan (C_{max}). Many approximate methods have been developed to overcome the limitations of exact enumeration techniques. These approximate approaches include simulated annealing (SA), tabu search and genetic algorithms (GA). However, real-world production systems require simultaneous achievement of multiple objective requirements. This means that the academic concentration of objectives in the JSP must been extended from single to multiple. Recent related JSP research with multiple objectives is summarized as below. Ponnambalam has offered a multi-objective GA to derive optimal machine-wise priority dispatching rules for resolving job-shop problems with objective functions that consider minimization of makespan, total tardiness, and total machine idle time. Ponnambalam's multi-objective genetic algorithm (MOGA) has been tested with various published benchmarks, and is capable of providing optimal or near-optimal solutions. One of the latest evolutionary techniques for unconstrained continuous optimization is particle swarm optimization (PSO) proposed by Kennedy et al. . PSO has been successfully used in different fields due to its ease of implementation and computational efficiency. Even so, application of PSO to the combination optimization problem is rare. Coello et al. provided an approach in which Pareto dominance is incorporated into PSO to allow the heuristic to handle problems with several object functions. The algorithm uses a secondary repository of particles to guide particle flight. That approach was validated using several test functions and metrics drawn from the standard literature on evolutionary multi-objective optimization. The results show that the approach is highly competitive. Liang et al. invented a novel PSO-based algorithm for JSPs. That algorithm effectively exploits the capability of distributed and parallel computing systems, with simulation results showing the possibility of high-quality solutions for typical benchmark problems. Lei presented a PSO for the multi-objective JSP to minimize makespan and total job tardiness simultaneously. Job-shop scheduling can be converted into a continuous optimization problem by constructing the corresponding relationship between a real vector and a chromosome obtained using the priority rule-based representation method. The global best position selection is combined with crowding-measure-based archive maintenance to design a Pareto archive PSO. That algorithm is capable of producing a number of high-quality Pareto optimal scheduling plans. Hybrid algorithms that combine different approaches to build on their strengths have led to another branch of research. Previous literature indicates that there has been little study of the JSP with multiple objectives. In this study, we use a new evolutionary PSO technique to solve the JSP with multiple objectives.

1.2 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 optimizes a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position and is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

2 LITERATURE REVIEW

2.1 Job constraints (β)

Brucker and T'Kindt and Billaut have done significant work in identifying various types of constraints on the job characteristics, which, when included, may significantly affect the realism of the scheduling model.
Yang and Liao define set-up times as the times of the tasks which need to be performed before a job can be processed immediately after another job on the same resource.

2.2 Availability intervals

Lee differentiates between three types of unavailability intervals, namely resumable,
non-resumable, and semi resumable unavailable intervals. White and Rogers address a job shop scheduling problem with unavailability intervals by means of a disjunctive graph formulation.

2.3 Auxiliary resources

Studies performed by Mason indicate that 16% of scheduled production cannot be met because tooling is typically not available. Additionally, 40% to 80% of a foreman's time is spent looking for and expediting materials and tools. Gargeya and Deane describe the multiple resource constrained job shop scheduling problem: a job shop in which two or more resource types constrain output. Brucker defines a class of problems that is very similar to the multiple resources constrained JSSP: the multiprocessor task job shop scheduling problem.

2.4 Objective function (γ)

The final field of Graham et al.'s notation focuses on this aspect of schedule optimization. Brucker provides a list of the most common measurements which can be used for objective function formulation. Any of the five measures in Table 2.1 can be used to formulate at least four different objective functions of the form: max $\{q_v \mid v = 1 . . . , n_j\}$, v=1njqv, v=1njwvqv and max$\{w_v q_v \mid v = 1 . . . , n_j\}$, where q_v denotes the measurement associated with job v, w_v and d_v denote the weight and the due date of job v, and n_j is the total number of jobs to be scheduled. For example, the following four objective functions can be formulated for job completion time, where C_v denotes the completion time of job v: make span (max $\{C_v \mid v = 1 . . . , n_j\}$), total flow time (v=1njCv) and weighted total flow time (v=1njwvCv).

Table 2.1: Commonly used job shop scheduling measurements

JSSP measurement	Formulation
Lateness	$L_v = C_v - d_v$
Earliness	$E_v = \max\{0, d_v - C_v\}$
Tardiness	$T_v = \max\{0, C_v - d_v\}$
Absolute deviation	$D_{1v} = \mid C_v - d_v \mid$
Squared deviation	$D_{2v} = (C_v - d_v)^2$

2.5 History and Early Methods of Optimization

One of the most famous and defining works in scheduling theory was published in the early 1960's. Giffler and Thompson's "Algorithms for Solving Production Scheduling Problems" introduced the most famous and widely used scheduling algorithm, called the GT algorithm. The GT algorithm insures the construction of an active schedule. Three years later "Industrial Scheduling" by Muth and Thompson was published which introduced the first famous job shop scheduling benchmark problem, a (10 ×10) problem that took 20 years to solve exactly, because of the high combinational complexity, which are the nature of such problems. In "A State-of-the-art Review of Job-Shop"Jain and Meeran divide the approaches to the solving this huge combinatorial problem into two main techniques, efficient and exact. Exact methods, obviously, use mathematical formulations to arrive exactly at the optimal solution. Since the problem space of the JSP problem is so large, these exact methods are not useful for anything but problems of small dimension, probably (10 ×10) or less. The proposed paper has focused on the application of a meta-heuristic technique called Particle Swarm Optimization (PSO). As will be shown, not much research has been conducted in the area of the JSP and the PSO, although the JSP does have a long history exploiting other meta-heuristic techniques shown in Figure, especially the Genetic Algorithm. Approximate methods, called approximate because they do not use mathematical formulations, calculation of gradient for example, to arrive at an exact optimal solution. In 1985 Davis was the first to apply an evolutionary algorithm in an effort to solve the Job Shop Problem. He used a genetic algorithm to evolve a priority list for each machine. An example of a priority list for a (5× 5) problem is given in Table 2.4

Table 2.2: Priority List Representation

	First Priority	Second Priority	Third Priority	Fourth Priority	Fifth Priority
Machine 1:	Job 2	Job 4	Job 1	Job 5	Job 3
Machine 2:	Job 1	Job 3	Job 5	Job 2	Job 4
Machine 3:	Job 4	Job 2	Job 5	Job 1	Job 3
Machine 4:	Job 3	Job 1	Job 2	Job 4	Job 5
Machine 5:	Job 5	Job 3	Job 4	Job 2	Job 2

Most of the research has been focused on the Genetic Algorithm (GA), although some consider the GA an ineffective way to solve the JSP. When compared to Simulated Annealing (SA), Tabu Search (TS), the GA seems to perform the poorest according to a comparative study done by Pirlot.

2.6 Particle Swarm Optimization

Particle Swarm Optimization As of yet, Particle Swarm Optimization has not been applied to the traditional Job Shop Problem, with one exception. In 2004 Weijun, et al. applied a hybrid Simulated Annealing/PSO to the traditional JSP. In their study, Simulated Annealing (SA) was used to fine tune solutions found by the PSO algorithm. The results of this hybrid algorithm were very promising. Tasgetiren, et al. in 2004 applied PSO to the Single Machine Weighted Tardiness problem. They developed the Smallest Value Position Rule (SVP) in order to transform the continuous space of the PSO to the permutation space used to represent a solution of the SMTWT problem. Tasgetiren also published a paper on the PSO applied to the Permutation Flowshop Sequencing Problem, along with Liang, Sevkli, and Gencyilmaz. The same SPV rule was used for the necessary space transformation from continuous to permutation space. This space transformation concept was used for the Traveling Salesman Problem by Pang, et al. in 2004. Their method of space transformation was called the GVP rule, or Greatest Value Priority. Using PSO and local search techniques they were able to solve medium scale (50 – 75 city) TSP Problems

Particle Swarm Optimization has also been applied to the Flexible Job Shop Problem (FJSP) by Xia and Wu recently in 2005.. They used the PSO not to determine the schedule, but to assign each operation to a machine as is required for the FJSP.
Not all researchers have used this particular space transformation technique to apply the PSO in permutation problems. In 2003 Pang, et al. developed a way to represent the difference in two permutations as a function of Swap Operators (SO). These Swap Operators perform a switch on two numbers in a permutation, and multiple swap operators in a given order form a Swap Sequence (SS). Lian, Gu and Jiao in 2005 developed a Similar Particle Swarm Optimization Agorithm (SPSOA) and applied it to the Flow-shop Problem (FSSP or PFSP).. These researchers, like Pang, et al. developed a way for the PSO algorithms to operate directly in the space of permutation problems. Their PSO algorithm is called "similar" because they used crossover and mutation techniques, easily performed on permutations, but originally developed for the Genetic Algorithms (GA). Their crossover and mutation operations were used to update the velocity and position of the particles in the PSO-like algorithm.

3 PROBLEM DEFINITION AND FORMULATION

3.1 The Job Shop Problem

The Job Shop Problem (JSP) is one of many types of scheduling problems that researchers from many fields are currently attempting to solve optimally using various meta-heuristic algorithms. The solution to these scheduling problems is simply the determination of the optimal assignment of a finite number of resources to a finite number of operations, while adhering to many pre-defined constraints, usually precedent constraints.

3.2 The Job Shop Problem Definition

A ($n \times m$) Job Shop Problem is defined by a specific number of jobs, n, each consisting of an order of operations, m, which are equal to the number of machines or resources specified in the problem. So a job, Ji is a predefined order of operations $\mathbf{O}_i = (O_{i,1}, O_{i,2}, \ldots, O_{i,m})$. Each operation O_{ij} has a processing time, or job duration, O_{ij}. For the traditional JSP the following rules apply:
• Each job must be processed by each machine in a certain order (precedent constraints)
• Each machine can only process one job at a time
• Each job can only be processed by one machine at a time
• Each job must be processed by each machine exactly once

• No preemption is allowed, or once a job has started processing it cannot be interrupted.

An example of a (10 ×10) JSP, the famous MT10 problem, is shown in table 3.1

Table 3.1: Scheduling Problem Example

	Machine Sequence (Processing Time)									
Job 1:	0 (29)	1 (78)	2 (9)	3 (36)	4 (49)	5 (11)	6 (62)	7 (56)	8 (44)	9 (21)
Job 2:	0 (43)	2 (90)	4 (75)	9 (11)	3 (69)	1 (28)	6 (46)	5 (46)	7 (72)	8 (30)
Job 3:	1 (91)	0 (85)	3 (39)	2 (74)	8 (90)	5 (10)	7 (12)	6 (89)	9 (45)	4 (33)
Job 4:	1 (81)	2 (95)	0 (71)	4 (99)	6 (9)	8 (52)	7 (85)	3 (98)	9 (22)	5 (43)
Job 5:	2 (14)	0 (6)	1 (22)	5 (61)	3 (26)	4 (69)	8 (21)	7 (49)	9 (72)	6 (53)
Job 6:	2 (84)	1 (2)	5 (52)	3 (95)	8 (48)	9 (72)	0 (47)	6 (65)	4 (6)	7 (25)
Job 7:	1 (46)	0 (37)	3 (61)	2 (13)	6 (32)	5 (21)	9 (32)	8 (89)	7 (30)	4 (55)
Job 8:	2 (31)	0 (86)	1 (46)	5 (74)	4 (32)	6 (88)	8 (19)	9 (48)	7 (36)	3 (79)
Job 9:	0 (76)	1 (69)	3 (76)	5 (51)	2 (85)	9 (11)	6 (40)	7 (89)	4 (26)	8 (74)
Job 10:	1 (85)	0 (13)	2 (61)	6 (7)	8 (64)	9 (76)	5 (47)	3 (52)	4 (90)	7 (45)

3.3 Related Scheduling Problems

The traditional Job Shop Problem has many "cousins", or other scheduling problems with the same goal, to produce an optimal schedule of a number of jobs through a number of machines. The Flexible Job Shop Scheduling Problem Another scheduling problem that proves it to be computationally hard is the Flexible Job Shop Scheduling problem (FJSP). The Flow Shop Scheduling Problem The Flow Shop Scheduling Problem (FSP) is another *n* job by *m* machine scheduling problem. The FSP differs from the JSP in that each job *j* has the same order of operations, or precedent constraints. The Single Machine Weighted Tardin**ess Problem** In the Single Machine Weighted Tardiness Problem (SMWTP) there is only one single machine and a list of operations to be processed on that machine.

3.4 Types of Schedules

Semi-active Schedules Semi-active schedules are schedules in which the next operation in a technological sequence is scheduled at the earliest allowable time. Active Schedules These are schedules in which no operation can be started earlier without violating a *precedent constraint*, or increasing the total processing time of any machine. Non-Delay Schedules These are schedules in which no machine is kept idle while it could be processing an operation. Parameterized Active Schedules Parameterized Active Schedules are non-delay schedules where the delay is *no more than* a specified parameter.

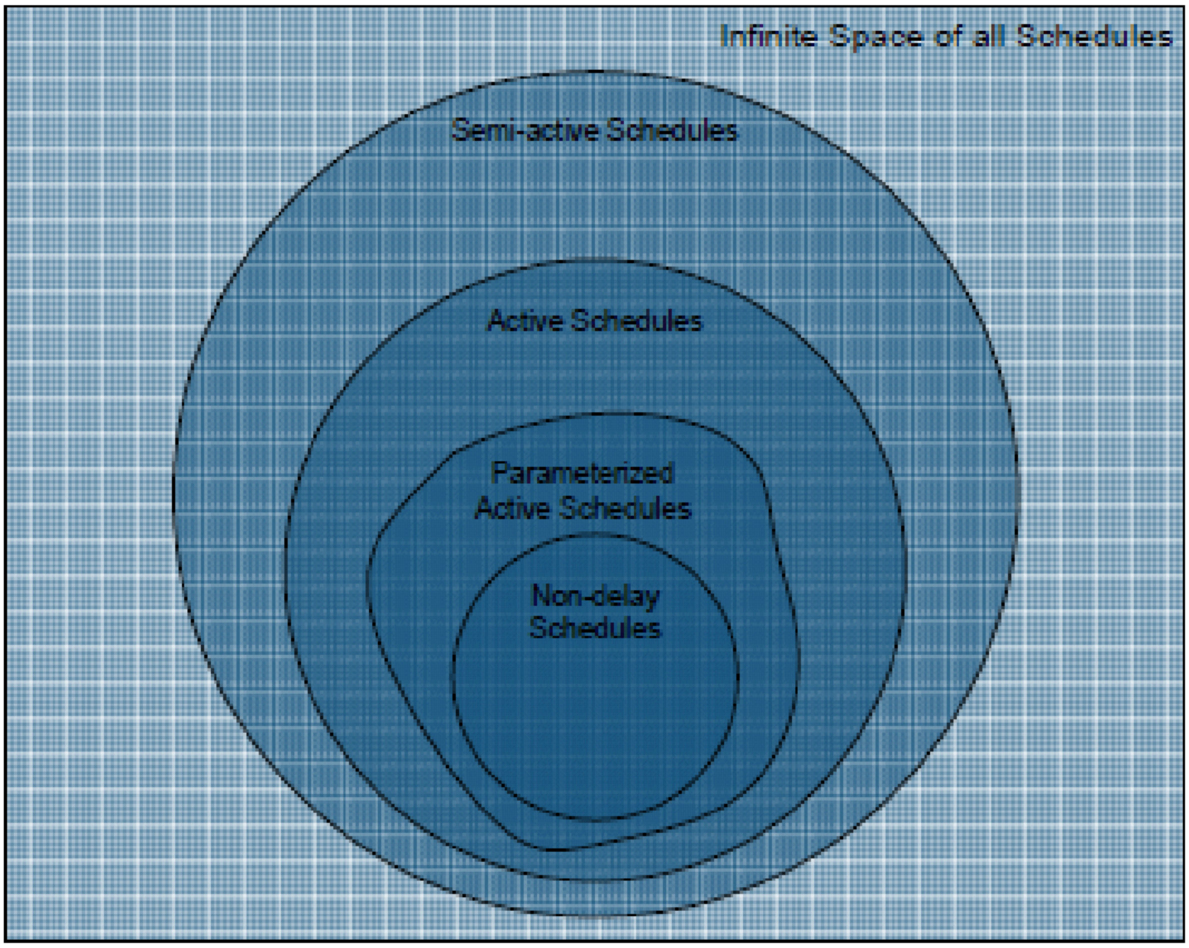
Infinite Space of all Schedules
Semi-active Schedules
Active Schedules
Parameterized Active Schedules
Non-delay Schedules

Fig 3.1 Relationship between Schedules

3.5 Representation of Schedules

In the examples of previous sections, Gantt Charts have been used to show various solutions to the Job Shop Problem. Gantt Charts allow a visual interpretation of a schedule, which is helpful when analyzing the makespan of a Job Shop Problem, or the classification of a schedule. Usually a Gantt chart is built from a *representation* of the schedule in the form of numbers or in the form of permutations. This representation can be of a direct fashion or indirect fashion, both have its advantages and disadvantages.

4 PROPOSED METHODOLOGY

The concept was to give each particle a social component and an individual component. Individual particles' behaviours would be influenced by their best positions (in the search space) found and by the best positions found by all particles in the swarm. Their hope was to design a search method that was able to find multiple optima not just the global. This way the particles can explore the search space and eventually converge to the global optimum. The agents or particles in this algorithm search the problem space by "moving through it" with a certain velocity. Each position a particle has in this space represents a possible solution to the problem at hand. The particles in a traditional PSO algorithm are governed by the following equations for a single dimension in Figure 4.1.

Momentum Component Social Component

$$v_i(t+1) = w * v_i(t) + c_1 * r_1 (pbest_i - x_i(t)) + c_2 * r_2 (gbest_i - x_i(t))$$

Velocity update equation

Personal Component

$$x_i(t+1) = x_i(t) + v_i\,(t+1)$$

Position update equation

- $i = 1, 2\ldots p$, p = number of particles in swarm
- pbest is the best position found by that particle so far
- gbest is the best position found by any particle so far
- v = velocity of particle in a single dimension
- x = position of particle in a single dimension
- t = iteration number
- w is the inertial constant
- c1, c2 are acceleration constants
- r1, r2 are random numbers evenly distributed between (0,1)

The problem space can be represented in a 3-dimensional cube. In order to optimize this problem using PSO, we must randomly generate particles in the search space. The size of our swarm will consist of only 3 particles in an effort to make the graphics that follow easier to understand. Normally, the swarm size is much larger, anywhere from the tens to the hundreds. The initial positions (representing three initial solutions) are shown in Figure 4.2 below. The constants C1 and C2 for this example will both be set to 2.
When those particles' values are plugged back into our objective function, we get the following:

Particle 1: 4.0759
Particle 2: 4.3770
Particle 3: 1.3088

Neither of these solutions is very good, however the best solution is the solution represent by Particle 3. Knowing how PSO works, we would expect the other two particles to move in the direction of Particle 3, at least at first. This can be seen in the figure below. After 100 iterations, the particles of travelled very close to the optimal solution. Their paths are shown in the next two figures, Figure 4.3 and 4.4 from different distances.

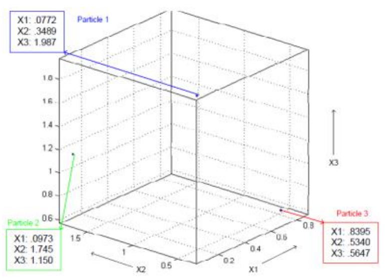

Fig. 4.2

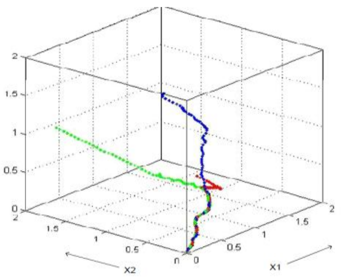

Fig 4.3

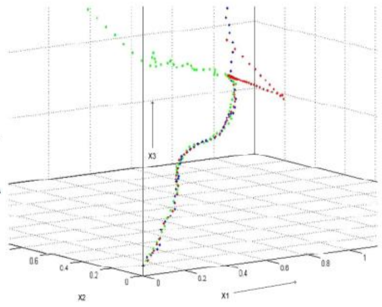

Fig 4.4

5 RESULT & DISCUSSION

The results of small size problem of 3*3 ie, 3 job on 3 machine, medium size problem of 6*6 ie, 6 job on 6 machine and a large problem of 10*10 ie, 10 jobs on 10 machine were predicted and it is found to be 12, 55 and 1194 seconds respectively. These results were obtained using particle swarm optimization (PSO) and it is better than results obtained by Genetic Algorithm. The machine processing time, idle time and make span time are also shown with the help of Gantt Chart.

6 CONCLUSION

Particle swarm optimization is an extremely simple algorithm that seems to be effective for optimizing a wide range of functions. We view it as a mid-level form of A-life or biologically derived algorithm, occupying the space in nature between evolutionary search, which requires eons, and neural processing, which occurs on the order of milliseconds. Social optimization occurs in the time frame of ordinary experience - in fact, it is ordinary experience. In addition to its ties with A-life, particle swarm optimization has obvious ties with evolutionary computation. Conceptually, it seems to lie somewhere between genetic algorithms and evolutionary programming. It is highly dependent on stochastic processes, like evolutionary programming.

7 SCOPE OF FUTURE WORK

Much further research remains to be conducted on this simple new concept and paradigm. The goals in developing it have been to keep it simple and robust, and we seem to have succeeded at that. The algorithm is written in a very few lines of code, and requires only specification of the problem and a few parameters in order to solve it. Further improvement can be done using hybrid optimization algorithms.

8 REFERENCES

[1] Bean, J., 1994. "Genetic algorithms and random keys for sequencing and optimization," Operations Research Society of America (ORSA) Journal on Computing, 6, 154–160.

[2] Beasley J.E., 1990. "OR-Library: distributing test problems by electronic mail", Journal of the Operational Research Society 41(11) pp1069-1072.

[3] Candido, M. A. B., Khator, S.K. & Barcia, R.M., 1998. "A genetic algorithm based procedure for more realistic job shop scheduling problems," International Journal of Production Research, 36(12), 3437–3457.

[4] Coello, C.A., Plido, G.T. & Lechga, M.S., 2004. "Handling multiple objectives with particle swarm optimization," IEEE Transactions on Evolutionary Computation, 8(3), 256–278.

[5] Esquivel, S.C., Ferrero, S.W. & Gallard, R.H., 2002. "Parameter settings and representations in Pareto-based optimization for job shop scheduling," Cybernetics and Systems: An international Journal, 33, 559–578.

[6] Fisher, H. & Thompson, G. L., 1963. Industrial Scheduling, Englewood Cliffs, NJ: Prentice-Hall.

[7] Garey, M. R., Johnson, D. S. & Sethi, R., 1976. "The complexity of flowshop and jobshop scheduling," Mathematics of Operations Research, 1, 117–129.

[8] Giffler, J. & Thompson, G. L., 1960. "Algorithms for solving production scheduling problems," Operations Research, 8, 487–503.

[9] Gonçalves, J. F., Mendes, J. J. M. & Resende, M. G. C., 2005. "A hybrid genetic algorithm for the job shop scheduling problem," European Journal of Operational Research, 167(1), 77–95.

[10] Jain, A.S. & Meeran, S., 1999. "Deterministic job-shop scheduling: Past, present and future," European Journal of Operational Research, 113, 390–434.

[11] Lei, D., 2008. "A Pareto archive particle swarm optimization for multi-objective job shop scheduling," Computers & Industrial Engineering, 54(4), 960–971.

[12] Liang Y.C., Ge, H.W., Zho, Y. & Guo, X.C., 2005. "A particle swarm optimization-based algorithm for job-shop scheduling problems," International Journal of Computational Methods, 2(3), 419–430.

[13] Lourenço, H. R., 1995. "Local optimization and the job-shop scheduling problem," European Journal of Operational Research, 83, 347–364.

[14] Nowicki, E. & Smutnicki, C., 1996. "A fast taboo search algorithm for the job shop problem," Management Science, 42(6), 797–813.