Optimization of Scheduling in Job shop Through Minimization of Makespan Using CPSO and GA

J.V.S. Bhaskar
Professor
Department of Mechanical Engineering, Sree Venkateswara College of Engineering
Nellore, Andhra Pradesh, India

Dr.B.Dattatraya Sarma
Principal
Sree Venkateswara College of Engineering, Nellore, Andhra Pradesh, India

Dr. K. Hema Chandra Reddy
Professor
Department of Mechanical Engineering, JNT UNIVERSITY
Anantapur, Andhra Pradesh, India

Abstract- Job shop scheduling is a very important concept that helps in maintaining efficiency and throughput. Job shop scheduling, being a NP-hard problem is challenging to optimize. In this paper a tool that can aid in scheduling of job shop is developed. This paper summarizes the results of the research work with an aim to effectively schedule job shop. In order to present the research to the user in the form of an automated tool, a Graphical User Inter phase (GUI) is designed and presented. This tool provides the user with ease of use and flexibility in choosing a particular schedule. The GUI is coded using MATLAB. Two metaheuristic approaches namely, Chaotic Particle Swarm Optimization (CPSO) and Genetic Algorithm (GA) are used to optimize the objective function to minimize makespan.This paper explains various features of the tool. The results of the tool are presented by analyzing different bench mark problems including large size problems typically comprising more than 50 jobs.

Keywords – M Job Shop Scheduling, CPSO, GA, MATLAB, Automatic tool

I. INTRODUCTION

Globalization and subsequent industrial reforms have brought challenges and opportunities alike to the door steps of Indian manufacturing sector. Indian industries have to adapt themselves to the changes in demand and the manufacturing process, management has to be effective in handling the available resources. Establishing an optimum schedule goes a long way in effectively managing the resourcing and improving productivity. Researchers have used various techniques that were developed under the general rubric of artificial intelligence to solve job shop scheduling problems [1]. Scheduling involves the allocation of resources over a period of time to perform a collection of tasks .It is a decision-making process that plays an important role in most manufacturing and service industries. Scheduling in the context of manufacturing systems refers to the determination of the sequence in which jobs are to be processed over the production stages, followed by the determination of the start-time and finish-time of processing of jobs. An effective schedule enables the industry to utilize its resources effectively and attain the strategic objectives as reflected in its production plan. The most common manufacturing system worldwide is the job shop. Job shops are associated with the production of small volume/large variety products and operate in make-to-order concept. Approximately 50 to 75 % of all manufactured components fall into this category of low volume/high variety and due to the market trends this percentage is likely to increase. Even though flexible manufacturing systems are today's keywords that frequently appear in many research agendas, scheduling of job shops still receive ample attention from both researchers and practitioners due to the reason that job shop scheduling problems exist in many forms in most of the advanced manufacturing systems. Besides, analysis of job shop scheduling problem provides important insights into the solution of the scheduling problems encountered in more realistic and complicated systems [2].

Scheduling has been a subject of a significant amount of literature in the operations research field since early 1950s [2] [3].The main objective of scheduling is an efficient allocation of shared resources over time to competing activities. Emphasis has been on investigating machine scheduling problems where jobs represent activities and machines represent resources. The problem is not only NP -hard, but also has a well-earned reputation of being one of the most computationally difficult combinatorial optimization problems considered to date. This intractability is one of the reasons why the problem has been so widely studied. The problem was initially tackled by "exact methods" such as the Branch And Bound method (BAB), which is based on the exhaustive enumeration of a restricted region of solutions containing exact optimal solutions. Exact methods are theoretically important and have been successfully applied to benchmark problems, but sometimes they are quite time consuming even for moderate-scale problems. With a rapid progress in computer technology, it has become even more important to find practically acceptable solutions by "approximation methods" especially for large-scale problems within a limited amount of time [3]. Stochastic local search methods are such approximation methods for combinatorial optimization. They provide robust approaches to obtain high quality solutions to problems of realistic sizes in reasonable amount of time. Some of stochastic local search methods are proposed in analogies with the processes in nature, such as statistical physics and biological evolution, and others are proposed in the artificial intelligence contexts. They often work as an iterative master process that guides and modifies the operations of subordinate heuristics; thus they are also called metaheuristics.

Metaheuristics have been applied to wide variety of combinatorial optimization problems with great success. Particle swarm optimization was developed by Kennedy and Eberhart (1995) as a stochastic optimization algorithm based on social simulation models. The algorithm employs a population of search points that moves stochastically in the search space. Concurrently, the best position ever attained by each individual, also called its experience, is retained in memory. This experience is then communicated to a part or to the whole population, biasing its movement towards the most promising regions detected so far. The communication scheme is determined by a fixed or adaptive social network that plays a crucial role as to the convergence properties of the algorithm [4]. The PSO algorithm searches for the best solution over the complex space through co-operation and competition. First of all, the PSO algorithm creates the initial particle swarm, namely, it initializes a swarm of particle randomly in the available solution space and function determines the fitness value through the target function. Each particle will move in the space of the solution, with its direction and distance determined by speed. The general particle will move following the best current particle, obtaining the best solution by searching generation by generation [5][6]. In each generation, the particle will trace two limited values, one of which is the best solution, pbest, which is found so far by the particle itself. The other is the best solution, gbest, which has been found so far by general group swarm [7]. Genetic algorithms belong to the class of evolutionary algorithms. These are algorithms which are based on the principles of natural evolution, and they can be divided into four major types of algorithms: genetic algorithms (GA), genetic programming, evolution strategies and evolutionary programming. All these types of algorithms are based on a population of individuals. Evolutionary algorithms have been applied to many problems in management, e.g., to location, inventory, production, scheduling, distribution or timetabling problems. The use of evolutionary algorithms for shop scheduling problems started around 1980. Two of the first applications to flow shop scheduling problems have been given by Werner [8, 9], and the first application to job shop scheduling problems can be found in [10]. Genetic algorithms are the most popular variant of evolutionary algorithms

In this paper we have presented a tool for job shop scheduling problem. The proposed tool is designed using Matlab. The tool is capable of optimizing scheduling using two metaheuristic optimization procedures like, Chaotic Particle Swarm Optimization (CPSO) and Genetic Algorithm (GA). The tool delivers the makespan value for that particular schedule as optimized by the chosen optimization approach. In order to provide the user with better visual perspective the Gantt chart of that particular schedule is also plotted. The tool also tabulates the idle times of each machine, thus enabling the user to identify which machine is having maximum idle time and thus can ensure better planning of work flow. The tool is expected to provide seamless user experience and ensure that better schedule is chosen for enhancing the throughput and efficiency.

II. JSP MATHEMATICAL MODEL

Shop scheduling problems belong to the class of multi-stage scheduling problems, where each job consists of a set of operations. Among the shop scheduling problems, there are three basic types: a flow-shop, a job shop and an open-shop. In a job shop problem, a specific technological route is given for each job. Scheduling is the allocation of shared resources over time to competing activities. The $n \times m$ job shop scheduling problem, designated by the symbols $n/m/G/C_{max}$ can be described by a set of n *jobs* $\{j_i\}_{1<j<n}$ which is to be processed on m machines $\{m_r\}_{1<r<m}$. Job shop scheduling problem shall meet the following constrained conditions:

(1)Each machine can only process one working procedure of certain work piece in a period of time.

(2)Each working procedure will not be interrupted by other working procedures during the processing.

(3)Each work piece shall experience the processing on m machines, and during the processing, new work piece shall not be added, and the processing cannot be terminated.

The objective of optimizing the problem is to find a schedule that minimizes C_{max}. .The objective function is defined as

$$c_{jr} = _{min} (c_{max})$$

$\{Cjr\}_{1\leq j\leq n;1\leq r\leq m}$ *is* the schedule of completion times for each operation that satisfies above constraints.C_{max} is the time required to complete all the jobs is called the makespan, where
$C_{max} = \max_{1\leq j\leq n;1\leq r\leq m} c_{jr}$.

The processing of job J_j on machine M_r is called the *operation* O_{jr}. Operation O_{jr} requires the exclusive use of M_r for an uninterrupted duration p_{jr}, its processing time; the preemption is not allowed. The starting time and the completion time of an operation O_{jr} is denoted as s_{jr} and c_{jr}
respectively. The predefined technological sequence of each job can be given collectively as a matrix $\{T_{jk}\}$ in which $T_{jk} = r$ corresponds to the k-th operation O_{jr} of job J_i on machine M_r.

III. CPSO

Particle swarm optimization (PSO) is a swarm intelligence algorithm that was put forward through the study on the bird swarm's flying behaviors. Similar to other optimization algorithm ideology, one particle denotes one bird in PSO, and each particle is assigned an initial location and speed. During the particle swarm flying process, flying speed and orientation will be constantly adjusted, so as to find the optimal solution [11]. In PSO algorithm, particle constantly updates its speed and location according to *best P* and *best g* . When one particle finds one optimal local solution, other particles will be attracted by the optimal solution to gather around the solution rapidly. That will lead to premature convergence and local optimization, which will accordingly influence PSO's search performance [12][13]. Each particle updates its position based upon its own best position, global best position among particles and its previous velocity vector according to the following equations:

$$v_i^{k+1} = w \times v_i^k + c_1 \times r_1 \times (p_{best_i} - x_i^k) + c_2 \times r_2 \times (g_{best} - x_i^k) \qquad (1)$$

$$x_i^{k+1} = x_i^k + \chi \times v_i^{k+1} \qquad (2)$$

Where,

v_i^{k+1} : The velocity of i^{th} particle at $(k+1)^{th}$ iteration
w : Inertia weight of the particle
v_i^k : The velocity of i^{th} particle at k^{th} iteration
$c_{1,}c_2$: Positive constants having values between [0, 2.5]

r_1, r_2 : Randomly generated numbers between [0, 1]

p_{best_i} :The best position of the i^{th} particle obtained based upon its own experience

g_{best} : Global best position of the particle in the population

x_i^{k+1} : The position of i^{th} particle at $(k+1)^{th}$ iteration

x_i^k : The position of i^{th} particle at k^{th} iteration

χ : Constriction factor. It may help insure convergence.

Suitable selection of inertia weight w provides good balance between global and local explorations.

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} \times iter$$

Where, $w_{\max}$ is the value of inertia weight at the beginning of iterations, $w_{\min}$ is the value of inertia weight at the end of iterations, $iter$ is the current iteration number and $iter_{\max}$ is the maximum number of iterations.

Chaotic is a nonlinear system that is similar to the "Random" and has complex behaviors. Since Chaotic is sensitive to the initial value, it can easily jump out of the local minimum. Also, its search speed is very fast. The basic ideology for CPSO algorithm is: In each iterative process, exert chaotic perturbation on *best g*, and take it as particle updating position, so as to prevent particle positions from converging, otherwise it will search locally around the global optimal solution. Block diagram of the CPSO method is shown in figure 1.

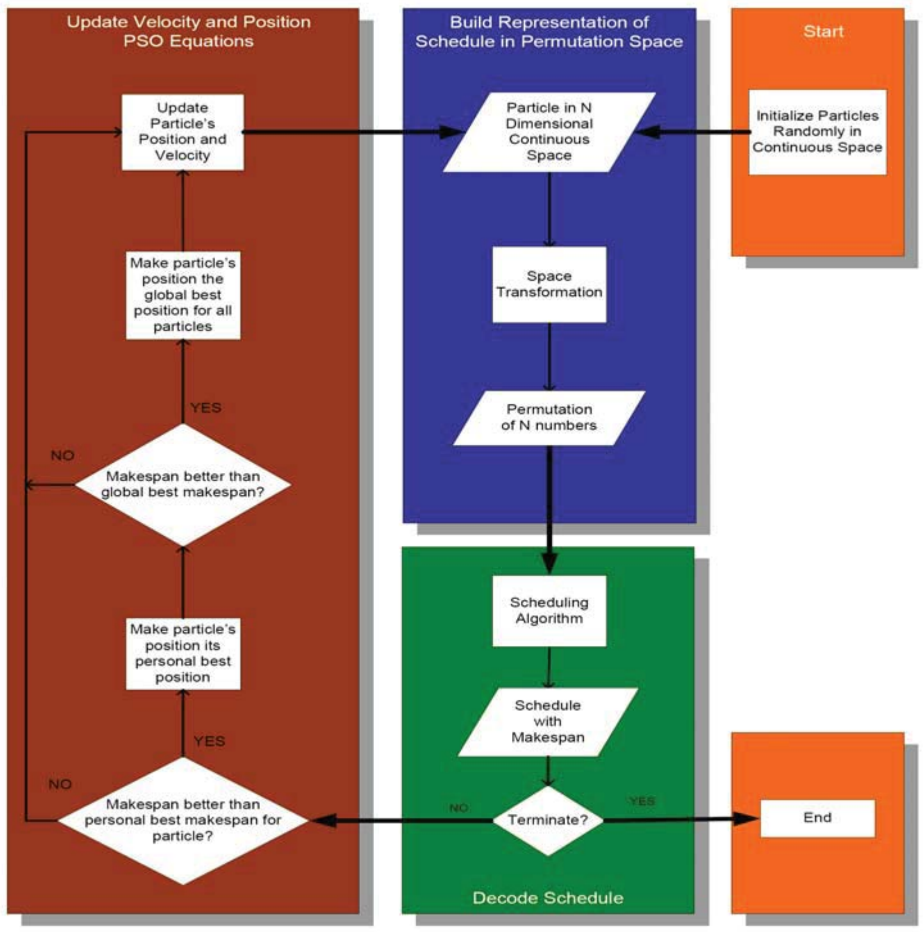

Update Velocity and Position PSO Equations
Build Representation of Schedule in Permutation Space
Start
Update Particle's Position and Velocity
Particle in N Dimensional Continuous Space
Initialize Particles Randomly in Continuous Space
Make particle's position the global best position for all particles
Space Transformation
Permutation of N numbers
YES
NO
Makespan better than global best makespan?
Scheduling Algorithm
Make particle's position its personal best position
Schedule with Makespan
YES
NO
Makespan better than personal best makespan for particle?
NO
Terminate?
YES
End
Decode Schedule

Figure 1: Block Diagram of the CPSO Method

IV. GENETIC ALGORITHMS

Genetic algorithm (GA) is one of the most widely used artificial intelligent techniques for optimization. GA was first developed by John Holland [14]. GA is stochastic searching algorithm based on the mechanisms of natural selection and genetics, and is very efficient in searching for global optimum solutions. The main idea of GA is to mimic the natural selection and the survival of the fittest [24]. In GA, the solutions are represented as chromosomes. The chromosomes are evaluated for fitness values and they are ranked from best to worst based on fitness value. The process to produce new solutions in GA is mimicking the natural selection of living organisms, and this process is accomplished through repeated applications of three genetic operators: selection, crossover, and mutation. First, the better chromosomes are selected to become parents to produce new offspring (new chromosomes) [14].The selection probabilities are usually defined using the relative ranking of the fitness values. Once the parent chromosomes are selected, the crossover operator combines the chromosomes of the parents to produce new offspring (perturbation of old solutions). Mutation is a mechanism to inject diversity into the population to avoid stagnation. In addition to the population size and the maximum number of iterations, several decisions on parameters must be made for GA. Crossover method and crossover probability are the second set of decisions to be made. Finally, the mutation method and mutation probability must be selected as they may help to maintain the diversity of the population by injecting new elements into the chromosomes. In general, these three sets of decisions are set empirically using pilot runs. The flow chart of the Genetic Algorithm is shown in Figure 2.

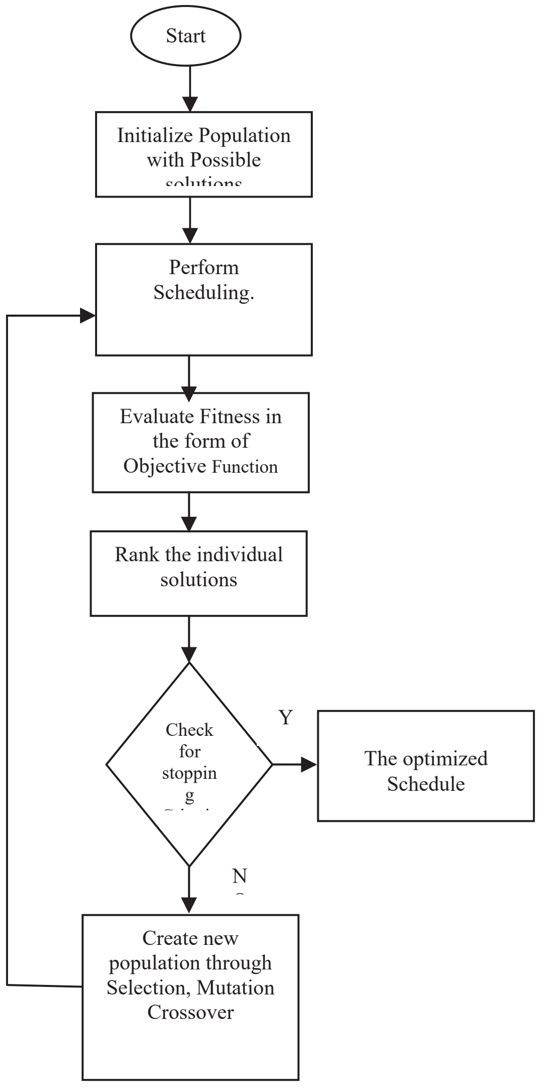

Start
Initialize Population with Possible solutions
Perform Scheduling.
Evaluate Fitness in the form of Objective Function
Rank the individual solutions
Check for stoppin g
Y
The optimized Schedule
N
Create new population through Selection, Mutation Crossover

Figure 2: The flow chart of the Genetic Algorithm for the proposed optimization problem

V. BENCH MARK PROBLEMS

The three well-known benchmark problems with sizes of 6×6, 10×10 and 20×5 (known as mt06, mt10 and mt20) formulated by Muth and Thompson [15] are commonly used as test beds to measure the effectiveness of a certain method. The mt10 and mt20 problems are almost similar. They are processing the same set of operations and technological sequences are similar, but in the mt20 problem, the number of machines available is reduced to half of that of the mt10 problem. For example, the first operation of each job in mt10 is exactly same as the first operation of each of the first 10 jobs in mt20 and the second operation of each job in mt10 is exactly same as the first operation of each of the second 10 jobs in mt20. Taillard proposed a set of 80 JSP and 120 FSP benchmark problems. They cover various ranges of sizes and difficulties. They are randomly generated by a simple algorithm. In this work 6 benchmark problems are considered namely mt06, mt10 and mt20 formulated by Muth and Thompson and Taillard's 15 jobs×15 Machines and 50 jobs×15 Machines, 100 jobs×20 Machines problems.

VI. GRAPHICAL USER INTERFACE (GUI)

One specific issue that is common to job shops across different industries is lack of availability of an efficient tool for scheduling operations. Some of the commercially available tools are either complex or capitally prohibited. Considering a job shop environment where most of the decisions happen at the floor level there is an utmost necessity for a simple and an efficient tool that can be help him solving the job shop process. Another issue that can be observed with commercially available tools is lack of ability to provide an easy interface to the user. Even though the problem of job shop has remained an active area of research it can be inferred much of the focus is towards the solving the problem rather than providing the solution to the end user. In order to deliver the results of research in a usable and a flexible platform a Graphical User Interface (GUI) has been designed and presented. This GUI encompasses the research in the form of a tool that enables the user to have a seamless use and flexibility of operation.

The functional structure of GUI is illustrated using the figure 3. The structure helps to understand the flow of data and the process in the GUI.

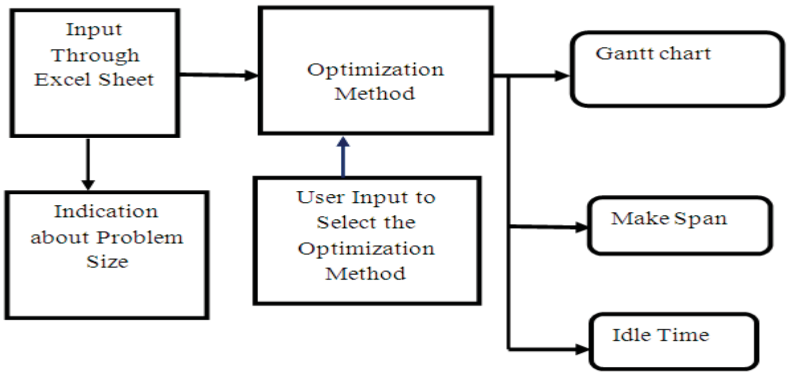
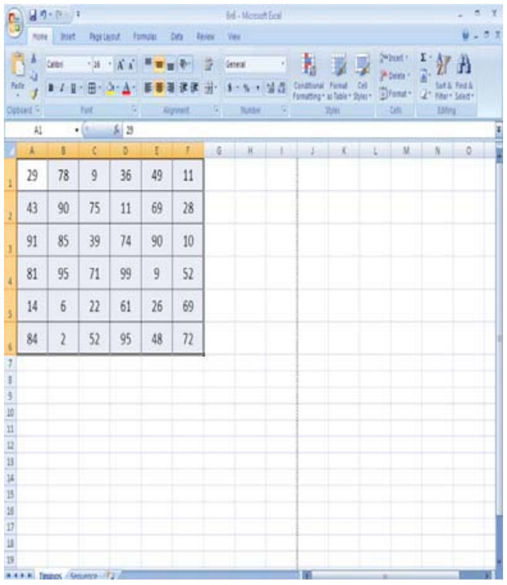

Input Through Excel Sheet
Optimization Method
Gantt chart
Indication about Problem Size
User Input to Select the Optimization Method
Make Span
Idle Time

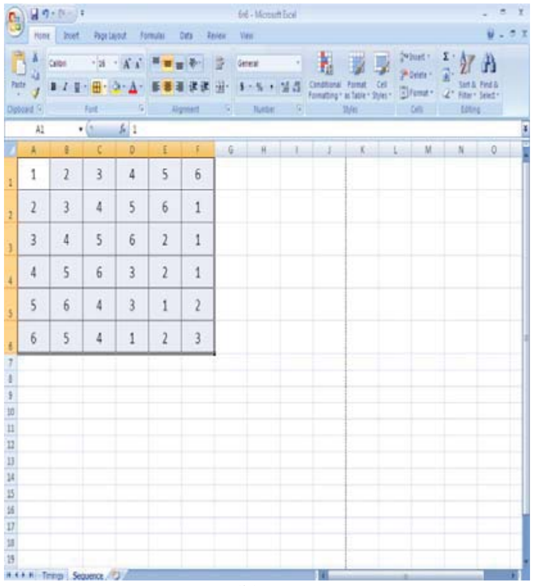
Figure 3: The functional structure of GUI

The necessary data for the optimization is fed through an excel file. The excel file is standardized to have two sheets, one representing the timing and other representing machining sequence. This
Standard format should help the user to input different problem size. This type of approach provides the user with enhanced flexibility that can help him in scheduling problems of different sizes. Figure 4 and represents Screen shot of Excel sheet used to give the sequence and timing input for 6×6 problem



29	78	9	36	49	11
43	90	75	11	69	28
91	85	39	74	90	10
81	95	71	99	9	52
14	6	22	61	26	69
84	2	52	95	48	72

4(a)



1	2	3	4	5	6
2	3	4	5	6	1
3	4	5	6	2	1
4	5	6	3	2	1
5	6	4	3	1	2
6	5	4	1	2	3

4(b)

Figure 4: Screen shot Excel sheet used to give the sequence and timing input for 6×6 problem

From the functional structure of GUI, it can be observed that the functional section of the tool can be classified into five different regions.

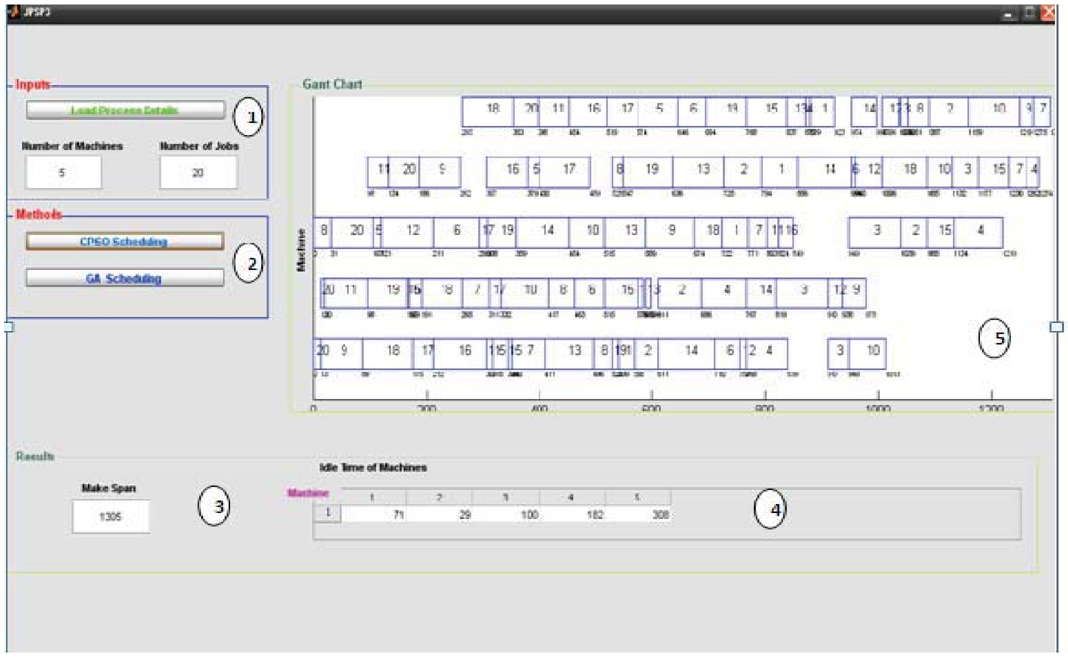
Section 1 helps the user in providing the inputs about the problem considered for the scheduling. The necessary data for optimization is fed through an excel file. The excel file is standardization to have two sheets with one sheet representing the timing of each operation and the other sheet representing the machining sequence. The standard sheet used for feeding the data to the tool for a 6 job 6 machine problems is given as example.

Once the user clicks the load process details functional icon the user is prompted to browse and select a particular excel sheet that has the process details. Once the data is loaded successfully a dialogue appears indicating the successful loading of data for further processing. Along with this there is a specific provision to display the number of jobs of that particular data being fed for optimization. This enables the user to have an initial tab on the problem size. This type of approach enables the user flexibility in choosing different problem sizes and provides the generic platform to analyze and optimize problems of different sizes.

Section 2 provides the necessary functional icons to choose different optimization methods for scheduling. This functional palette currently holds two functional icons representing optimization through CPSO and GA. There is scope to add more methods in the future. Once the user chooses the particular optimization approach the program initiates scheduling through that approach and optimizes the scheduling operation within the defined constraints of optimization. Once a scheduling operation is successfully implemented a dialogue appears indicating the successful completion of optimization. Simultaneously results related to sections 3, 4, and 5 are displayed.
Section 3 displays the makespan of the particular schedule. This section provides the minimum makespan for a particular schedule as obtained through a specific optimization approach. The makespan indicator is specific to the best schedule obtained by the optimization procedure for that particular trial run.

Section 4 displays the idle time experienced by each machine for that particular schedule. The indication of idle time provides a comprehensive view about the delay experienced by different machines for that particular schedule. It also helps in visualization of idle time as percentage of makespan. This gives a clear indication about individual machine utilization and total machine utilization.

Section 5 displays the Gantt chart in the user interface and also a separate image is opened in which a specific Gantt chart is displayed. Using this display the user will be able to store that specific image for future analysis. The Gantt chart being a effective visualization tool of a particular schedule can be a great help in visualization of the process.



Number of Machines: 5

Number of Jobs: 20

Results

Make Span: 1306

Idle Time of Machines

Machine	1	2	3	4	5
1	71	29	100	182	308

Figure 5: Screen Shot of the GUI Designed

The functional icons present in the GUI can be described as below in reference to the Figure 5.

1) Functional icon used to load the data, in the form of Excel sheet which has the machine sequence and timing details
2) This functional icon is used to choose different optimization method for scheduling
3) The makespan of that particular schedule is displayed here
4) Idle time of machines is displayed here.
5) Gantt chart of the schedule

VII. RESULTS AND CONCLUSION

Genetic Algorithm (GA) function available in the Matlab optimization tool box is used in this work. The population size is fixed at 20. The elite count used is fixed at 10 % of the population which is 2. The selection is based on ranking. The cross over fraction is fixed at 0.2 and the adaptive feasible mutation function is used. The migration of the population is fixed as forward with a forward fraction of 0.2. The maximum number of generations is fixed at 100. The iteration settings for CPSO include 100 maximum numbers of iterations, with acceleration constant of 2 and 2.5 and maximum and minimum inertia weights at 1 and 0.2 respectively. The maximum and minimum velocity of particles is fixed at 0.003 and -0.003 respectively. The simulations are carried out in a system having Core 2 Duo processor cloaking a speed of 2 GHz with a RAM of 2GB

The Results of makespan achieved by two different optimization methods of CPSO and GA are given in the Table 1.

Table 1: Makespan as achieved by the two Optimization Methods of CPSO and GA

Problem Size	Makespan Using CPSO	Makespan Using GA	% Reduction in Makespan for GA
mt06(6 jobs ×6 Machines)	58	55	5.17
mt10(10 jobs ×10 Machines)	1046	938	10.32
mt20(20 jobs × 5 Machines)	1289	1180	8.45
Taillards 15 jobs × 15 Machines	1578	1299	17.68
Taillards 50 jobs × 15 Machines	3991	3299	17.33
Taillards 100 jobs×20 Machines	8456	6642	21.45

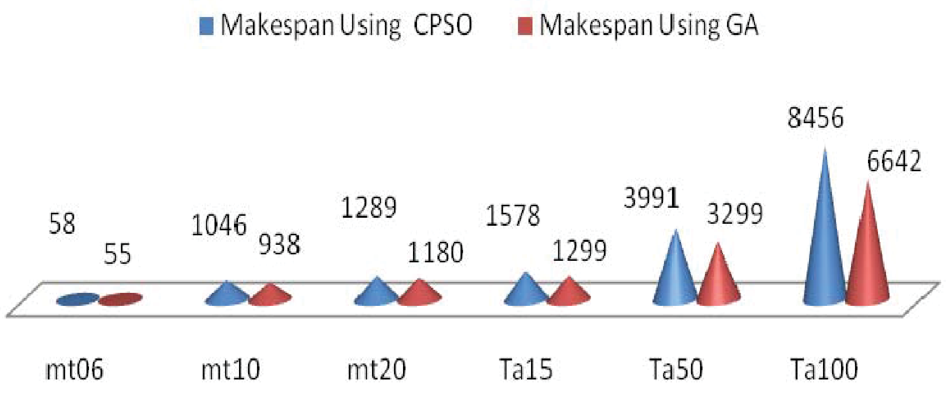


Figure 6: Plot of Makespan for different problem sizes and optimization methods.

The results are best results achieved when each optimization method is run for 50 times. It can be observed from Table 1 and figure 6 the scheduling results produced by GA performs better than CPSO. When compared to the CPSO based optimization the GA based optimization reduces the makespan by nearly 5.17 % for the Mt06 problem. There is a similar reduction of 10.32 % for the Mt10 problem which comprises 10 machines and 10 jobs. There is a significant 8.45 % reduction in the makespan between CPSO and GA for Mt20 problem. The optimization for Taillards problem also sees a significant reduction in the makespan between CPSO and GA. The makespan arrived at by GA is less than 17.68 % than that produced by CPSO for a Tail lards 15 jobs×15 Machines problem. Similarly in the case of Tail lards 50 jobs×15 Machines there is reduction of 17.33 % for GA when compared to the makespan produced by CPSO. Similarly for Taillards 100 jobs × 20 Machines there is a reduction of 21.45 %

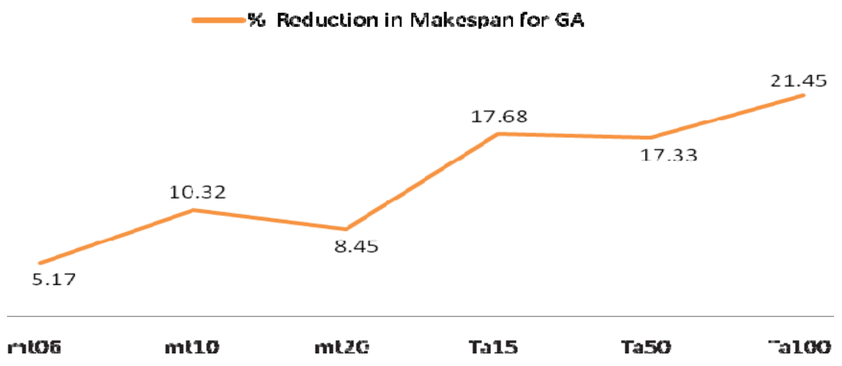


Figure 7: Plot of % reduction in makespan for GA over CPSO

The program also computes the idle time experienced by the machines when the scheduling is done using different optimization approaches. The idle time values give an insight in to the efficiency of the machines and the utilization of each machine. This data will be particularly helpful in identifying the machine redundancies as well.

Table 2: mt 6jobs×6Machines - Idle Time

Method	Makespan	Idle Time of Machines					
		M1	M2	M3	M4	M5	M6
CPSO	58	17	14	21	37	19	14
GA	55	11	6	17	34	15	11

The table 2 provides an illustration of idle times experienced by different machines when the operation is scheduled by CPSO and GA. It can be inferred from the table 2 that the maximum idle time is experienced by Machine M4 in both the cases and the minimum idle time is experienced by Machine M2.

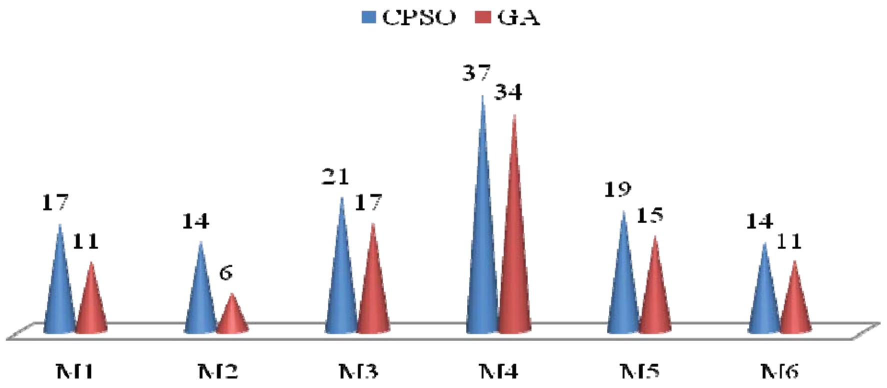


Figure 8: Plot of mt 6 jobs × 6 Machines - Idle Time

Analysis of machine idle time is important from the perspective of understanding the process flow and improving the machine utilization. The optimized scheduled as optimized by GA has improved the machine utilization by minimizing the machine idle time. Table 3 illustrates the maximum idle time experienced by machines for different bench marks when the schedule is arrived at using different optimization approaches of CPSO and GA.

Table 3: Maximum idle time experienced by machines in different test problems

Problem Size	Maximum Idle Time (on Machine) CPSO	Maximum Idle Time (on Machine) GA
mt06 - 6 jobs × 6 Machines	37 (M4)	34(M4)
mt10 -10 jobs × 10 Machines	618(M10)	559(M10)
mt20 -20 jobs × 5 Machines	308(M5)	184(M5)
Tail lards 15 jobs × 15 Machines	971(M10)	692(M10)
Tail lards 50 jobs × 15 Machines	1799(M6)	1038(M6)
Tail lards 100 jobs × 20 Machines	3718(M10)	1932(M10)

It can be observed from the table 3 there is significant reduction in maximum idle time experienced by a machine for a given schedule. Both the methods converge in indentifying the machine which is experiencing maximum idle time. It can also be observed that there is a significant reduction in maximum idle time experienced by the machine when the schedule is optimized using GA. Figure 9 depicts the percentage reduction in maximum idle time for machines when the scheduled is optimized using GA instead of CPSO.

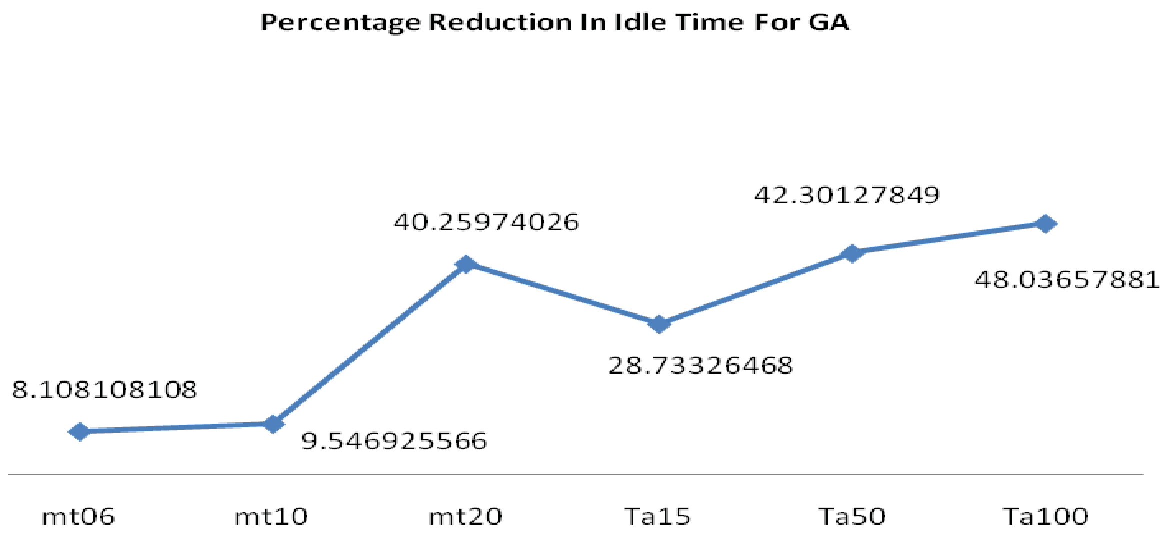


Figure 9: Plot of Maximum idle time experienced by machines

A series of trial runs were executed to identify the best, worst and average makespan time delivered by optimization approaches GA and CPSO. The Results of this analysis are as illustrated below. The results of mt06 (6 jobs×6 Machines) is illustrated in figure 10.

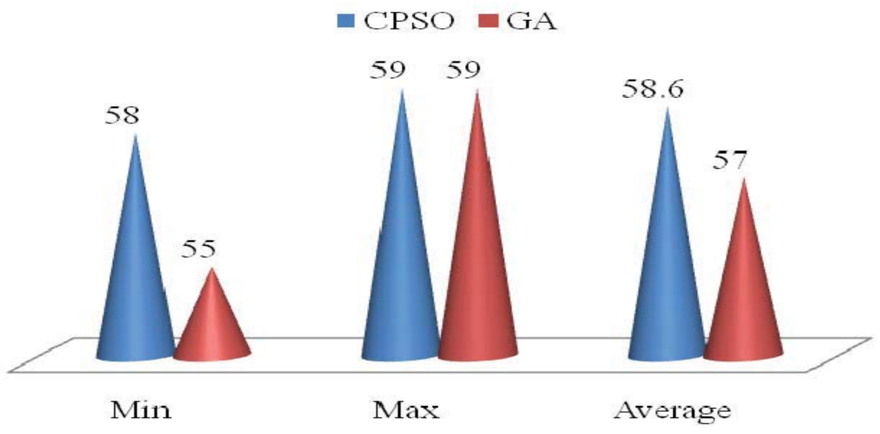


Figure 10: Minimum, Maximum and Average idle times delivered by CPSO and GA for Mt06

The results of analysis for mt10 (10 jobs×10 Machines) is depicted in the figure 11, it can be observed that the performance of GA is better in comparison to CPSO for all the three aspects. It can also be inferred that in terms of minimum makespan and average makespan delivered by GA its performance is much better when compared to CPS O.

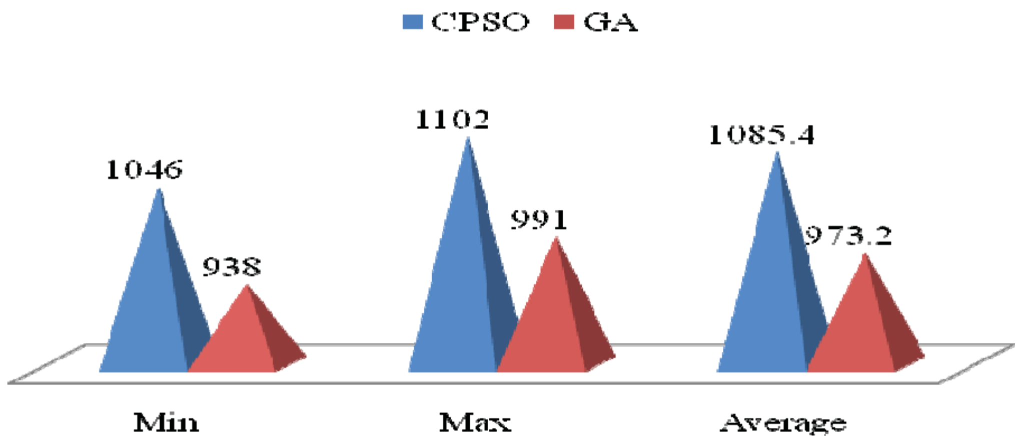


Figure 11: Minimum, Maximum and Average idle times delivered by CPSO and GA for mt10

Similarly as an illustration for a large problem size, analysis for Taillard's 100 jobs×20 Machines is given in the figure 12. For this case also it can be observed that the performance of GA is relatively better than the performance of CPSO. The minimum makespan delivered by GA is 1814 units lesser than that produced by CPSO which translates to 27.2% lesser makespan for GA when compared with CPSO.

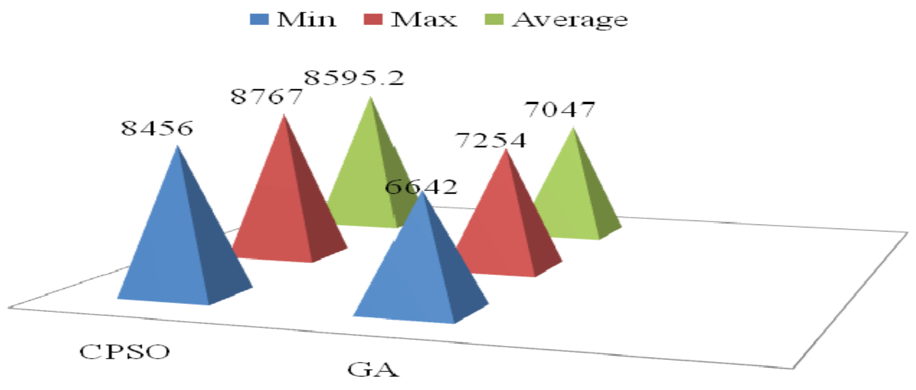


Figure 12: Minimum, Maximum and Average idle times delivered by CPSO and GA for Taillard's 100 jobs×20 Machines

Gantt chart [19] is one typical representation in bar chart that is used to represent a feasible schedule in job shop scheduling problem. A Gantt chart is a convenient way of visually representing a solution of the job shop scheduling problem. In this work the results of the scheduling operation are also presented in the form of a Gantt chart drawn automatically through the developed interface. The figure 13 gives the Gantt chart of the optimized schedule as suggested by genetic algorithm for Mt 06 problem.

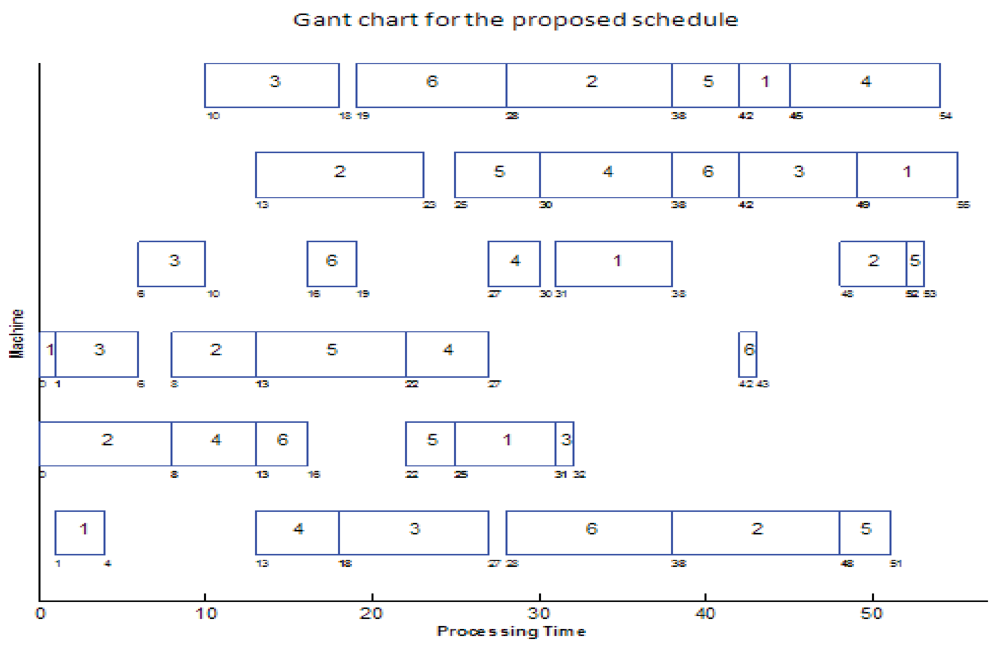


Figure 13: Gantt chart as plotted by the GUI for mt 06 Problem

Scheduling is a very important aspect in effective utilization of resources and increasing throughput in manufacturing industry. In this work we have designed a tool for scheduling using CPSO and GA. The tool is comprehensive in that it is capable of providing a seamless integration to the user. Apart from providing the makespan for a particular schedule the tool is capable of populating the idle time experienced by each machine for that specified schedule. The Gantt chart is drawn automatically for each optimized schedule. This helps in visualisation, interpretation and provides better understanding of the results. From the optimization standpoint it can observed that the GA based optimization performs better when compared to the CPSO optimization. The machine idle times experienced by the machines helps in identifying the redundancies and improve machine utilization. Similarly the Gantt chart helps in providing a visual insight to have improved understanding of the effective schedule.

REFERENCES

[1] Lee C.Y, Piramuthu S, Tsai Y.K, “Job shop scheduling with a Genetic algorithm and machine learning” International Journal of production research, vol 4 (1997) pp 48-56.

[2] French, S. (1982), “Sequencing and Scheduling: An introduction to the mathematics of Job-Shop”, Ellis Horwood Limited, Chichester.

[3] Shinn-Ying Ho, Hung-Sui Lin, Weei-Hurng Liauh, Shinn-Jang Ho, “Orthogonal particle swarm optimization and its application to task assignment problems”, IEEE Transaction on Systems,Man, and Cybernetics-part A: Systems and Humans, vol.38, no.2, pp.288-298, 2008.

[4] Shu Wang, Ling Chen, “A Particle Swarm Optimization Algorithm Based on Orthogonal Test Design”, Fifth International Conference on Natural Computation, pp.190-194, 2009.

[5] R. C. Ebenhart and Y. Shi, “Particle Swarm Optimization: Developments, Applications and Resources” Evolutionary Computation, vol.1, 2001, pp. 81–86

[6] Feng Pan, Xiaohui Hu, Russ Eberhart, “An Analysis of Bare Bones Particle Swarm”, IEEE Swarm Intelligence Symposium, pp.1-5, 2008.

[7] Qingfu Zhang, Yiu-Wing Leung, “An orthogonal genetic algorithm for multimedia multicast routing”, IEEE Transaction on Evolutionary Computation, vol. 3, no.1, pp.53-62, 1999.

[8] Werner, F. On the solution of special sequencing problems. Ph.D. Thesis, TU Magdeburg, 1984 .

[9] Werner, F. An adaptive stochastic search procedure for special scheduling problems. Economicko-Matematicky Obzor, 1988, 24, 50 – 67.

[10] Davis, L. Job Shop scheduling with genetic algorithms. In: Grefenstette (ed.), Proceedings of the First International Conference on Genetic Algorithms and Their Applications. Lawrence Erlbaum: Hillsdale, NJ, 1985, 136 – 140.

[11] M. Senthil Arumugam, M.V.C. Rao, Alan W.C. Tan, “A novel and effective particle swarm optimization like algorithm with extrapolation technique”, Applied Soft Computing, no.9,pp.308–320, 2009.

[12] J.F. Muth and G.L. Thompson. *Industrial Scheduling*. Prentice-Hall, Englewood Cli_s,N.J., 1963.

[13] Jie Yang, Abdesselam Bouzerdoum, Son Lam Phung, “A particle swarm optimization algorithm based on orthogonal design”, IEEE Congress on Evolutionary Computation, pp. 1-7, 2010.

[14] Matlab R 2012 a Optimization Tool Box Reference Manual

[15] Zhi-hui Zhan, Jun Zhang, Ou Liu, “Orthogonal Learning Particle Swarm Optimization”,Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pp.1763-1764, 2009.

[16] Xiaomei YI, Peng WU, Dan DAI, Lijuan LIU, Xiong HE, "Intrusion Detection Using BP Optimized by PSO", IJACT, Vol. 4, No. 2, pp. 268 ~ 274, 2012

[17] Chen Lei, "A Hierarchical PSO Algorithm for Self-organizing Neural Network Design", AISS,Vol. 4, No. 1, pp. 132 ~ 138, 2012
[18] T.Yamaguchi, K.Yasuda, “Adaptive particle swarm optimization: Self-coordinating mechanism with updating information”, IEEE International Conference on Systems, Man and Cybernetics,pp.2303–2308, 2006.
[19] James M. Wilson, “Gantt charts: A centenary appreciation”, European Journal of Operational Research 149 (2003) 430–437.