International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 19 (2015) pp 40165-40170

An Automated Tool for Job Shop Scheduling by Optimisation of Make Span through CPSO and GA

J. V. S. BHASKAR

Professor, Sree Venkateswara college of Engineering, Nellore, A. P India Jvsphd@gmail. com

Dr. B. Dattatreya sarma

Principal Sree Venkateswara College of Engineering, A. P, India, bdsarma@rediffmail. com

Dr. K. Hema chendra reddy,

Professor, JNTUA UNIVERSITY ANANTAPUR, A. P, India konireddy@gmail. com

Abstract

The job shop scheduling problem is one of the most difficult NP hard combinatorial optimization problems. Owing to the high computational complexity, it is quite difficult to achieve an optimal solution with traditional optimization approaches. This paper explores job shop scheduling to reduce makespan through Particle Swarm Chaotic Particle Swarm Optimization (CPSO) and Genetic Algorithm (GA) approaches. A Graphical User Interface (GUI) is designed to automate the steps involved in optimization of scheduling. The GUI provides the user flexibility in handling different machine systems and seamlessly integrates with decision making process. The results of the two optimization approaches are compared and Gantt chart of the schedule plotted.

Keywords : Job shop scheduling, Makespan, Genetic Algorithm Chaotic Particle Swarm Optimization, Graphical User Interface.

1. Introduction

Job shop scheduling has been proven to be an intractable problem for analytical procedures. Recent advances in computing technology, especially in artificial intelligence, have alleviated this problem by intelligently restricting the search space considered, thus opening the possibility of obtaining better results. 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 volumes/large variety products and operate in make-to-order mentions that 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 problems 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 the 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, making each particle an available solution of the optimization problem. Furthermore, the target 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

2. 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*m* jobshop 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 *Cmax*.. The objective function is defined as $\boldsymbol{c_{jr} = min\ (c_{max})}$
$\{Cjr\}1\leq j\leq n;1\leq r\leq m$ *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 $\boldsymbol{C}max = \max 1\leq j\leq n;1\leq r\leq mcjr$.

The processing of job *Jj* on machine *Mr* is called the *operation Ojr*. Operation *Ojr* requires the exclusive use of *Mr* for an uninterrupted duration *pjr*, its processing time; the preemption is not allowed. The starting time and the completion time of an operation *Ojr* is denoted as *s jr* and *cjr* 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 *Ojr* of job *Ji* on machine *Mr*.

3. PSO and 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. A detailed block diagram of the proposed work is given in the figure below.

Figure (1): Block Diagram of the proposed CPSO Method

4. 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(GA) is shown in Figure (2).

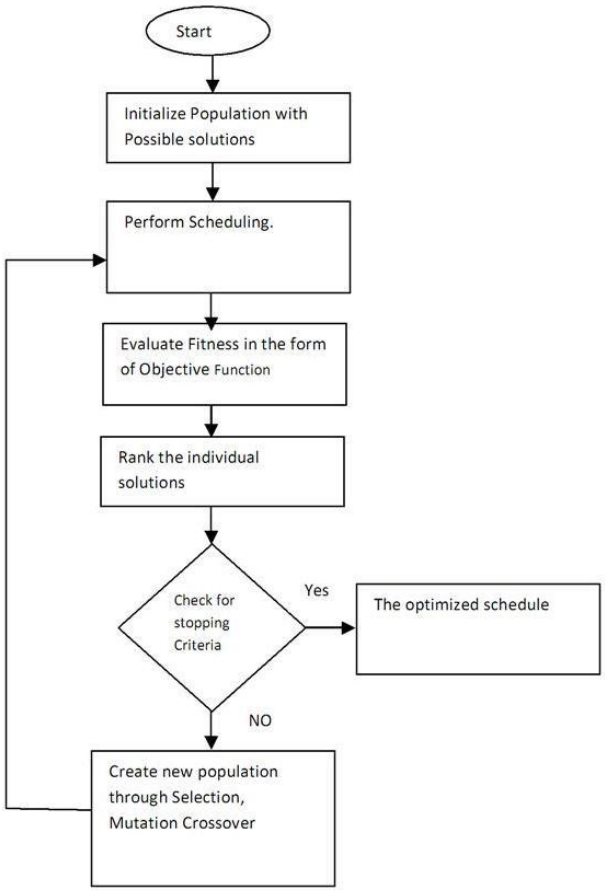

Figure (2): The flow chart of the Genetic Algorithm for the proposed optimization problem

5. 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 5 benchmark problems are considered namely mt06, mt10 and mt20 formulated by Moth and Thompson and Taillard's 15 jobs * 15 Machines and 30 jobs * 15 Machines problems.

6. Matlab Graphical User Interface (GUI)

MATLAB is widely used in all areas of applied mathematics, in education and research at universities, and in the industry. MATLAB stands for MATrix LABoratory and the software is built up around vectors and matrices. This makes the software particularly useful for linear algebra but MATLAB is also a great tool for solving algebraic and differential equations and for numerical integration. MATLAB has powerful graphic tools and can produce nice pictures in both 2D and 3D. It is also a programming language, and is one of the easiest programming languages for writing mathematical programs. MATLAB also has some tool boxes useful for signal processing, image processing, optimization; etc. MATLAB has many advantages compared to conventional computer languages for solving technical problems. MATLAB is an interactive system whose basic data element is an *array* that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide.

The GUI is coded using MATLAB Version 7. 1 and is designed to enable the user to have seamless analysis of the data using different optimization methods. The data required for analysis is fed through an Excel sheet in predefined format. This helps in standardizing the input methods and helps in avoiding user induced errors. A Graphical User Interface enables the user to have seamless use and flexibility of operation. The implementation is carried out in a system having Core 2 Duo processor cloaking at a speed of 2 GHz with a RAM of 2GB.

Figure (3): Screen Shot of the GUI Designed

From the GUI it can be observed that Input to the system can be given by pressing the Load Process Details switch. Once the data is loaded the number of machines and the number of jobs involved in the scheduling are displayed below. The user can then choose the required method for scheduling. The scheduling results are displayed in terms of make span of each proposed schedule, a decision table which tabulates the job sequence and a Gantt chart.

The functional icons present in the GUI can be described as below in reference to the Figure (3).

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) Machine and the job sequence is displayed here
5) Gantt chart of the schedule

7. Results and Conclusion

In this work Genetic Algorithm (GA) function available in the Matlab optimization tool box is used in the proposed work. The population size is fixed at 20. The elite count used is fixed at 10 % of the population which 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 make span achieved by different optimization methods of PSO, CPSO and GA are given in the Table 1.

Table 1: Make Span as achieved by the Optimization Methods.

Problem Size	Make Span using PSO	Make Span Using CPSO	Make Span Using GA
Mt06- 6 jobs * 6 Machines	59	57	54
Mt10- 10 jobs * 10 Machines	1036	972	954
Mt20- 20 jobs * 5 Machines	1328	1296	1205
Tail lards 15 jobs * 15 Machines	1435	1384	1317
Tail lards 30 jobs * 15 Machines	2975	2765	2673

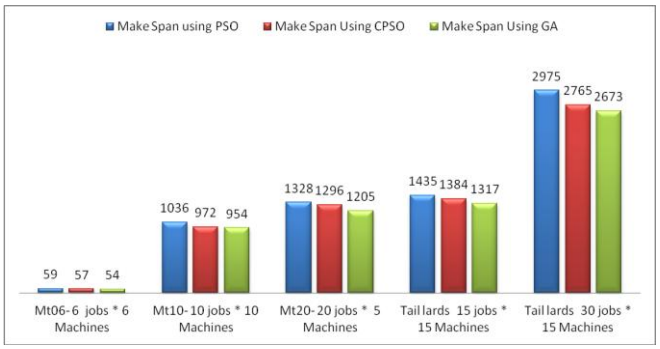

Figure (4): Plot of Makespan for different problem sizes and optimization methods.

The results are best results achieved when each optimization method is run for 50 times. It can be observed from the above Table and figure (3) the scheduling results produced by GA outperforms both PSO and CPSO. When compared to the PSO based optimization the GA based optimization reduces the makespan by nearly 8 % for the Mt06 problem. There is a similar reduction of 8 % for the Mt-10 problem which comprises 10 machines and 10 jobs. There is a significant 9. 2% reduction in the makespan between PSO and GA for Mt - 20 problem. The optimization for Taillards problem also see a significant reduction in the makespan between PSO and GA.

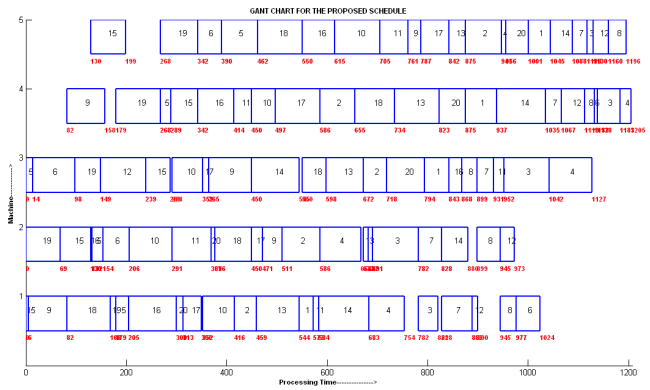

Figure (5): Gantt chart as plotted by the GUI for Mt 20 Problem

The above figure gives the Gantt chart of the optimized schedule as suggested by genetic algorithm for Mt 20 problem in which 20 jobs are processed across 5 machines.

7. 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 ransaction 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 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 andG. 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.