An Ant Colony Optimization algorithm for Flexible Job shop scheduling problem

S. G. Ponnambalam[1], N. Jawahar[2] and B. S. Girish[3]
[1]*School of Engineering, Monash University*
[2]*Thiagarajar College of Engineering, Madurai*
[3]*National Institute of Technology, Calicut*
[1]*Malaysia*
[2,3]*India*

1. Introduction

Scheduling involves the allocation of resources over a period of time to perform a collection of tasks (Baker, 1974). It is a decision-making process that plays an important role in most manufacturing and service industries (Pinedo, 2005). 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 (Conway et al., 1967). 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 a make-to-order environment (Groover, 2003). Hoitomt et al. (1993) 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 (Kutanoglu & Sabuncuoglu, 1999). Besides, analysis of job shop scheduling problems provides important insights into the solution of the scheduling problems encountered in more realistic and complicated systems (Pinedo 2005). In this context, this chapter focuses on scheduling job shops which is an important task for manufacturing industry in terms of improving machine utilization or reducing lead time.

1.1 Classical Job shop scheduling problem

The classical job shop scheduling problem (JSP) is the most popular scheduling model in practice (French, 1982; Brucker, 1995; Pinedo, 1995). It has attracted many researchers due to its wide applicability and inherent difficulty (Jain & Meeran, 1999). The formulation of the JSP is based on the assumption that for each part type or production order (job) there is only one processing plan, which prescribes the sequence of operations and the machine on which each operation has to be performed. The *n* x *m* classical JSP involves *n* jobs and *m* machines. Each job is to be processed on each machine in a predefined sequence and each machine processing only one job at a time. It is also well known that JSP is NP-hard (Garey et al., 1976).

1.2 Scheduling Job shops associated with multiple routings

In practice, the shop-floor setup in a job shop typically consists of multiple copies of the most critical machines so that bottlenecks due to long operations or busy machines can be reduced (Ho et al., 2007). Therefore, an operation may be performed on more than one machine. Job shops also consists of multipurpose machines such as numerically controlled (NC) machines that are loaded with tool magazines and are capable of performing several different types of operations (Vairaktarakis & Cai, 2003). Due to the overlapping capabilities of these machines, a given operation can be performed by more than one machine. However, in real life it has been a practice that machining operations are assigned to a certain machine tool during the process planning stage and the assignment of machine tools over time to different operations is performed during the scheduling stage. Recently, researchers considered the integration of process planning with scheduling by allowing alternative machine tool routings for operations at the scheduling stage.

Hankins et al. (1984) discussed the advantages of using alternative machine tool routings to improve the productivity. They showed, through an example, that using alternative machine results in reduced lead-time and improves overall machine utilization. Chryssolouris and Chan (1985) discussed the integration of process planning and the decision making process for production resource assignment. They discussed the issue of generating alternative machines/resources based on the process planning information. Wilhelm and Shin (1985) investigated the effectiveness of alternative operations in flexible manufacturing systems. They showed via computational experiments that alternative operations could reduce flow time while increasing machine utilization. However, the consideration of alternative routing option in job shop adds an additional decision of machine allocation during scheduling that increases the complexity of the problem. Therefore, scheduling of job shops that are associated with multiple routings is a much more complex problem than the JSP. Even though the practical applications and advantages of using multiple routings in job shop scheduling are more when compared with fixed routing, still the research focus in this area is very limited. On the above considerations, this chapter addresses a scheduling model of the job shop problem associated with multiple routings.

Two different scheduling models of job shop associated with multiple routings are addressed in the literature. The first model is referred as job shop scheduling with alternative machine tool routings, which was first addressed by Iwata et al. (1978). The same model was later addressed by Brandimarte (1993) as flexible job shop scheduling problem (FJSP). The second model is usually referred as job shop scheduling with multi-purpose machines (MPM-JSP), which was first addressed by Brucker and Schlie (1990). Dauzere-Peres and Paulli (1997) addressed the MPM-JSP as multiprocessor job shop scheduling problem (MJS). The difference between the two models (FJSP and MPM-JSP/MJS) is that, in the first model the processing time for each operation on its alternative routes differs with machine features, whereas in the second model the processing time is same for all the alternative machines of a particular operation. Since the FJSP can be represented as a generalized model of MPM-JSP/MJS, therefore, many recently published research articles refer both the models as FJSP. This chapter, therefore, considers FJSP as the scheduling model of the multiple routing job shop problem and proposes an Ant Colony Optimization based heuristic to solve the problem.

1.3 Review of solution methodologies

Since FJSP belongs to the category of NP-hard problems, therefore, heuristic methods had been a predominant choice for the researchers over the traditional mathematical techniques. The heuristic approaches for solving FJSP are generally classified as hierarchical and integrated approaches. In hierarchical approaches, assignment of operations to machines and the sequencing of operations on the machines are treated separately, i.e. assignment and sequencing are considered independently. In integrated approaches assignment and sequencing are not differentiated and considered together.

Iwata et al. (1980) presented two dispatching rules for the FJSP. The first rule is based on EFT (Earliest Finishing Time) rule in which the activity with the earliest finishing time from a set of competing activities is assigned first to a machine tool. The EFT rule is extended to EFTA (Earliest Finishing Time with Alternative Operations) rule to solve the job shop problem with alternative operations. Nasr and Elsayed (1990) investigated the problem of minimizing the mean flow time in a general job shop type machining system with alternative machine tool routings. They developed a Mixed-Integer Linear Programming (MILP) formulation for the problem and proposed two hierarchical approaches to solve the problem. The first heuristic solves the problem by decomposing the problem into sub-problems that are easier to solve. The second heuristic called Shortest Finish Time (SFT) rule is an extension of the first heuristic and is based on the Shortest Processing Time (SPT) rule.

During the past two decades there has been much research on the application of metaheuristics to solve the FJSP. The early research on FJSP was focused on development of neighborhood based metaheuristics like Tabu Search (TS) and Simulated Annealing (SA) algorithms. Brandimarte (1993) proposed a hierarchical approach based on TS to solve the FJSP with the objective of minimizing the makespan time. Hurink et al. (1994) represented the FJSP as a disjunctive graph model and proposed a hierarchical approach based on TS to solve the problem. Dauzere-Peres and Paulli (1997) presented a new disjunctive graph model to represent the MJS problem and proposed an integrated approach to solve FJSP. Mastrolilli and Gamberdella (2000) improved the TS approach proposed by Dauzere-Peres and Paulli (1997) and presented two new neighbourhood functions to solve FJSP instances. Najid et al. (2002) proposed a modified SA method for solving FJSP for minimum makespan time criterion. They represented their problem as a disjunctive graph and used the neighbourhood functions developed by Mastrolilli and Gamberdella (2000). Recently, Mehrabad and Fattahi (2005) presented a tabu search algorithm that solves the FJSP with sequence dependent setups to minimize the makespan time. They compared the performance of the algorithm with the optimal solution obtained using a MILP model solved by lingo software. Fattahi et al. (2007) developed a MILP model for the FJSP for minimum makespan time criterion and solved using Lingo software. They showed that solving FJSP using Lingo (Branch and Bound method) is very time consuming and are suitable for solving only smaller size problems. They also proposed two set of heuristics to solve the real size problems in which one set of heuristics is based on integrated approach and the other set is based on hierarchical approach. In integrated approach, they used Tabu Search (TS) and Simulated Annealing (SA) heuristics and presented two algorithms. In hierarchical approach, they used TS and SA and proposed four algorithms. Though neighborhood based metaheuristics have been successfully applied to solve FJSP, still the performance of these heuristics depend upon the initial solution and are more susceptible of getting stuck in local optimum. Therefore, most of the recently published research articles on FJSP are focused on developing Population based metaheuristics like Genetic Algorithm (GA), Ant Colony Optimization (ACO), etc. that are useful for any hard optimization problem. Mesghouni et al. (1998) were the first to model GA for FJSP. They proposed a chromosomal representation known as parallel job representation in which a chromosome is represented by a matrix where each row consists of a set of ordered operations of each job. Due to the complexity of decoding the representation, their algorithm incurs significant computational cost. Hussain and Joshi (1998) proposed a two pass GA to solve job shop problem with alternative routing with the objective of minimizing the sum of squared weighted due date deviation for every job. The first pass picks the alternatives using a genetic algorithm and the second pass provides the order and start time of jobs on the selected alternatives by solving a non-linear program. Chen et al. (1999) proposed a GA that uses an A-B string representation to solve FJSP for minimum makespan time criterion. A string contains a list of all operations of all jobs and the machines selected for the corresponding operations while B string contains a list of operations that are processed on each machine. Moon and Lee (2000) developed a mixed integer linear programming (MILP) model and proposed a GA for the job shop scheduling problem with alternative routings. The objective they considered is to minimize the mean flow time. The chromosome representation in their proposed GA consists of two strings, one for machine assignment and the other for schedule generation. Ho and Tay (2004) proposed a GA based tool, namely GENACE, for solving the FJSP for minimum makespan time criterion. The chromosome representation consists of two components, one component for machine selection and the other for operation sequence. Their methodology first generates an initial population using composite dispatching rules. A cultural evolution is then applied to preserve knowledge of schemata and resource allocations learned over each generation. The knowledge or belief spaces in turn influence mutation and selection of individuals. Ho et al. (2007) proposed an architecture for learning and evolving of flexible job shop schedules for minimum makespan criterion called learnable genetic architecture (LEGA), a generalization of their previous approach GENACE (Ho and Tay, 2004). Their proposed LEGA architecture is functionally divided into three modules: namely, a population generator module, Evolutionary algorithm (EA) module and a Schemata Learning (SL) module. The chromosome representation consists of two components, one component for machine selection and the other for operation sequence. The population generator module generates a set of feasible schedules equal to the population size using Composite Dispatching Rules and then encodes it into chromosomes of initial population for subsequent evolution in the EA module. During genetic evolution, the SL module modifies the offspring schedules to improve solution quality and to preserve feasibility based on a memory of conserved schemas resolved from sampled schedules sent dynamically from EA module. Tay and Ho (2008) proposed a genetic programming (GP) based approach for evolving effective composite dispatching rules for solving the multi-objective FJSP. The objective they considered is to minimize the weighted sum of makespan time, mean flow time and mean tardiness. They proposed a GP framework in which an individual is composed of terminals (like job release dates, due date, processing time, current time, remaining time, etc.) and algebraic functions. Their GP solves a specific problem by carefully selecting the terminals and functions and generating a composite dispatching rule that satisfies the requirements of that particular problem. They generated five composite dispatching rules using a large training set and compared the results with other popular rules like FIFO, SPT, etc.

In recent years, researchers have shown that ACO, which is a kind of metaheuristic search approach, is competitive with other approaches in terms of performance and CPU requirements in several applications of general and combinatorial optimisation problems. Ant algorithms were first proposed by Dorigo et al. (1992) as a multi-agent heuristic search approach to solve the traveling salesman problem (TSP). Dorigo and Stutzle (1999) suggested various ACO approaches to quadratic assignment problems (QAP), e.g. ant system (AS), ACS, ANTS–QAP, MAX–MIN ant system (MMAS), FANT, and HAS–QAP and compared the results. There is currently considerable activity in the scientific community devoted to extending/applying ant-based algorithms to many different discrete optimisation problems. Recent applications cover problems such as vehicle routing, sequential ordering, graph colouring, routing in communications networks, etc. In the field of scheduling, ACO has been successfully applied to the single machine weighted tardiness problem (den Besten et al., 2000; Liao & Juan, 2007), flow shop scheduling problem (Rajendran & Ziegler, 2004; Gajpal & Rajendran, 2006) and the resource constrained project scheduling problem (Merkle et al., 2002). Colorni et al. (1994) were the first to apply ACO to tackle the job shop scheduling problem (JSP). Blum (2002) proposed an MMAS algorithm for the Group shop scheduling problem which is a generalization of the JSP and the open shop scheduling problem. However, the performances of these algorithms for JSP were far from reaching the state-of-the-art performance. Blum and Sampels (2004) proposed MMAS algorithm with hyper cubic framework for the group shop scheduling problem. They performed experiments with different visibility functions and ranked them based on their performance. They used non-delay schedule generation mechanism for constructing solutions and employed a local search procedure to improve the solutions. Their algorithm performs particularly well for open shop scheduling problems. Heinonen and Pettersson (2007) proposed a hybrid ACO-local search algorithm for solving JSP. They performed experiments with different visibility functions on four different ACO variants and showed that MMAS algorithm performs better than AS, Rank based–AS and ACS. Huang and Liao (2008) presented a hybrid algorithm combining ACO algorithm with a taboo search algorithm for the JSP. Their proposed ACO algorithm employs a decomposition method inspired by the shifting bottleneck procedure, and a mechanism of occasional re-optimizations of partial schedules. The taboo search algorithm is embedded in the ACO algorithm to improve the solution quality. Rossi and Dini (2007) proposed an ACO based software system for solving flexible job shop problem (FJSP) with routing flexibility, sequence-dependent set up and transportation time. They showed the effectiveness of their system by comparing with other alternative approaches (including GA) for various benchmark instances. Girish and Jawahar (2008) proposed MMAS based heuristic for the FJSP for minimum makespan time criterion. Their proposed algorithm outperformed a GA and a constraint programming model solved using ILOG Solver for various benchmark problem instances.

1.4 Summary

The literature review on FJSP reveals that population-based search heuristics such as GA and ACO have emerged as powerful tools for solving FJSP in the recent years. The coding schemes adopted in most of the proposed GAs for FJSP requires repair mechanisms to maintain solution feasibility. A few researchers have incorporated certain schemes to maintain the feasibility of solution which restricts the search to a smaller solution space. Most of the GAs proposed, therefore, have chances of missing the best optimal solution even under extensive searches for larger size problems. The literature review on ACO for scheduling applications reveals that ACO is competitive with other metaheuristic approaches including GA in terms of computational time and solution quality.

In light of the above, this chapter proposes an ACO algorithm to solve the problem under discussion for minimization of makespan time criterion. The proposed ACO algorithm is based on the MMAS algorithm proposed by Stützle and Hoos (2000). The proposed ACO incorporates pheromone trails for both route choice option and active feasible schedule generation and is capable to provide all possible instances that an enumerative search can. The proposed ACO is structured and coded in such a way that it can be easily adapted to generate schedules for any scheduling objective of FJSP.

The rest of the chapter is organized as follows: Section 2 describes the problem; the proposed ACO is explained and illustrated in section 3; Section 4 presents a numerical illustration for the proposed ACO; Section 5 presents the performance study of the proposed ACO for various benchmark instances and Section 6 concludes with scope for future work

2. Description of the problem

2.1 Problem environment

- There are *m* machines in the system and *n* jobs to be processed.
- Each job *i* requires J_i precedence-constrained operations to be performed.
- Each operation *j* of job *i* (O_{ij}) can be processed on a number of alternative (non-identical) machines and its processing time t_{ijk} differs with the features of machine *k*. This addresses the multiple routings for jobs. An alternative routing could be used if one machine tool is temporarily overloaded while another is idle. The alternative routing is useful where capacity problem arises.

2.2 Assumptions

- Jobs are independent and no priorities are assigned to any job type.
- Job pre-emption or cancellation is not allowed.
- Set up and inspection times are included in the processing time.
- All jobs are simultaneously available at time zero.
- After a job is processed on a machine it is transported to the next machine immediately and the transportation time is negligible or included in the operation time.
- Breakdowns are not considered.

2.3 Objective

The objective is to complete all operations at the earliest possible time, which is known as minimum makespan time. This objective would distribute the workload evenly among all processing stations or work centers and all the processing stations would be freed at the makespan time for planning another set of jobs of the next planning horizon.

2.4 Problem definition

Scheduling in such an environment requires the following two types of decisions:

- Assignment of each operation to one of its alternative routes (machines) so as to convert the multiple routing job shop problem to a fixed route job shop problem.
- Sequencing the operations on the machines and generating start time and finish time for each operation.

Therefore, the problem can be defined as:

> “Determination of optimal or near optimal schedules for the flexible job shop problem by assigning the operations to one of its alternative machines and sequencing the operations on the machines for the objective of minimization of makespan time, given the processing time of operations on all its alternative routes and the precedence relationship between the operations”.

2.5 Problem formulation

The mathematical formulation for the problem under discussion with the objective of minimizing makespan time is presented below:
Objective:

$$\text{Minimize } [\text{Max}(C_{1J_1}, C_{2J_2} ..., C_{nJn})] \tag{1}$$

Subject to:

$$C_{ij} - S_{ij} - \sum_{\{k:Oij \in N_k\}} (t_{ijk} \,.\, X_{ijk}) = 0 \quad \forall \quad i, j \tag{2}$$

$$\begin{aligned} &C_{i'j'} - C_{ij} + H(1 - Y_{iji'j'k}) + H(1 - X_{ijk}) + H(1 - X_{i'j'k}) \geq t_{i'j'k}, \\ &\quad \forall \quad k, (i,j),(i',j') : O_{ij} \in N_k, O_{i'j'} \in N_k \end{aligned} \tag{3}$$

$$\begin{aligned} &C_{ij} - C_{i'j'} + H(Y_{iji'j'k}) + H(1 - X_{ijk}) + H(1 - X_{i'j'k}) \geq t_{ijk}, \\ &\quad \forall \quad k, (i,j),(i',j') : O_{ij} \in N_k, O_{i'j'} \in N_k \end{aligned} \tag{4}$$

$$S_{ij} \geq 0, \quad \forall \quad i, j \tag{5}$$

$$S_{ij+1} - C_{ij} \geq 0, \quad \forall\, i, j = 1,..., J_i - 1 \tag{6}$$

$$\sum_{k:\,Oij\in N_k} X_{ijk} = 1, \quad \forall\, i,\, j \tag{7}$$

$$X_{ijk} = \begin{cases} 1,\text{if operation } O_{ij} \text{ is assigned to machine } k \\ 0,\text{ otherwise} \end{cases} \tag{8}$$

$$Y_{iji'j'k} = \begin{cases} 1,\text{ if operation } O_{ij} \text{ precedes } O_{i'j'} \text{ on machine } k \\ 0,\text{otherwise} \end{cases} \tag{9}$$

where, S_{ij} and C_{ij} is the start time and completion time of job i, H is a very large positive integer, N_k is the set of operations $\{O_{ij}\}$ that can be loaded on machine k, X_{ijk} is a decision variable for machine selection for operation O_{ij} and $Y_{iji'j'k}$ is a decision variable that generates a sequence between the operations O_{ij} and $O_{i'j'}$ for loading on machine k. The constraints set (2) imposes that the difference between the completion time and the starting time of an operation is equal to its processing time on the machine to which it is assigned. This constraint satisfies the assumption that once an operation has started, it cannot be pre-empted until its completion. Constraints set (3) and (4) ensure that no two operations can be processed simultaneously on the same machine. This disjunctive constraints set (3) becomes inactive when $Y_{iji'j'k}$=0 and the disjunctive constraints set (4) becomes inactive when $Y_{iji'j'k}$=1. Constraints set (5) ensure that the start time of an operation is always positive. Constraints set (6) represent the precedence relationship among various operations of a job. Constraints set (7) impose that an operation can only be assigned to one machine.

3. Description of the proposed ACO

The different modules of the proposed ACO for the flexible job shop problem are outlined as flowchart in Figure 1. The different modules are described below.

Input module: The following data pertaining to the problem are given as input: Number of Jobs (n), number of machines in the shop (m), number of operations J_i of each job i ($\forall i$), number of alternative machines (routes) R_{ij} for operation j of job i ($\forall i$, $\forall j$), the machine number K_{ijr} corresponding to the route r of operation j of job i along with its processing time T_{ijr} ($\forall i$, $\forall j$, $\forall r$).

Initialization module: The number of ants (*no_ant*) is defined, and the pheromone trails used by them for constructing solutions are initialized. This problem uses two pheromone trails: pheromone trail intensity for route selection $\tau_{ijr}(tn)$ gives information about the desirability of choosing route r for operation O_{ij} at iteration tn and pheromone trail intensity $\varepsilon_{kiji'j'}(tn)$, which indicates the desirability of choosing operation O_{ij} directly after the operation $O_{i'j'}$ is loaded on machine k, is used for job conflict resolution while generating feasible schedule using Giffler and Thompson procedure. Thus, $\varepsilon_{kiji'j'}$(tn) indicates the pheromone trail between the operations. The pheromone trails $\tau_{ijr}(1)$ and $\varepsilon_{kiji'j'}(1)$ are initialized to the upper trail limit to $\tau_{max}(1)$ and $\varepsilon_{max}(1)$, respectively, which causes a higher exploration at the start of the algorithm.

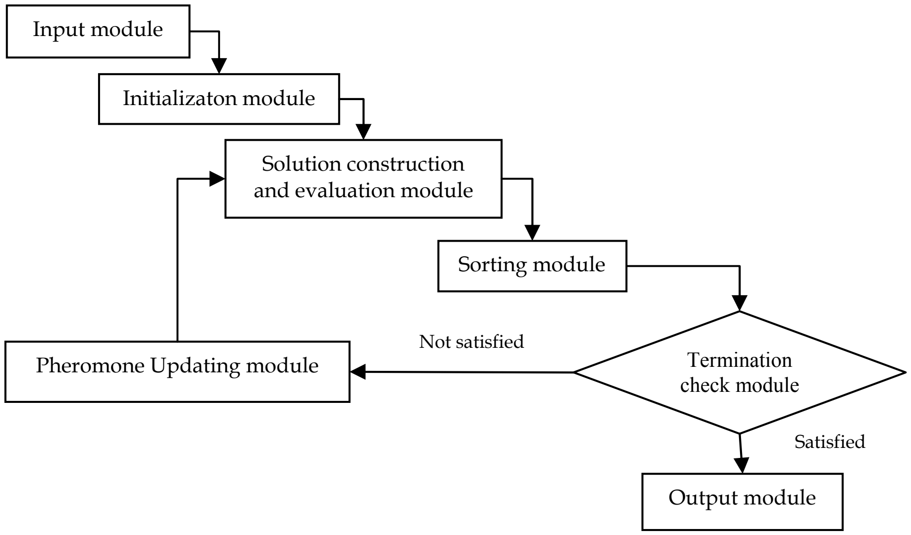

Fig. 1. Procedure of the proposed ACO for FJSP

Solution construction and Evaluation module: Each ant constructs a solution in two stages. In the Ist stage, an ant, at each construction step, allocates an operation of a particular job to one of its available resources. The ants use a probabilistic choice rule which is a function of the pheromone trail $\tau_{ijr}(tn)$ and a heuristic information based on processing time. Ant z chooses to allocate operation O_{ij} to the route r with a probability given by

$$P'^{Z}_{ijr}(tn) = \frac{[\tau_{ijr}(tn)]^{a}.[\eta_{ijr}]^{\beta}}{\sum_{l=1}^{R_{ij}}[\tau_{ijl}(tn)]^{a}.[\eta_{ijl}]^{\beta}} \tag{10}$$

where a and β are two parameters that control the relative importance of the pheromone trail and the heuristic information (visibility of ant). η_{ijr} represents the heuristic information which is a function of the processing time of operation O_{ij} on route r and is given by $\eta_{ijr}=1/T_{ijr}$. Therefore the probability is a trade-off between visibility (which says that operations requiring lower processing time will have a higher probability to be loaded on the machines, thus implementing a greedy constructive heuristic) and trail intensity at iteration tn.

In the IInd stage, on allocation of all operations to the machines, each ant generates a schedule based on Giffler and Thompson (1960) algorithm. Ants, at each construction step s, chooses an operation O_{ij} to allocate to its machine that has the minimum earliest finishing time ($EFT_i(s)$) among the unassigned operations, provided the chosen operation has no conflict with other operations on the same machine. Ants store the operations that are assigned to its resources in Q_k^z (s) at each step s. Any conflict, if arises in the process of schedule generation is resolved using a probabilistic choice rule which is a function of pheromone trail intensity and a heuristic function. If a conflict arises in the construction step *s* of an ant *z*, then it resolves the conflict by finding the probability of loading operation O_{ij} to its resource *k* directly after operation $O_{i'j'}$ loaded on the same machine. If O_{ij} is the first operation in the sequence, then $O_{i'j'}=0$. The probabilistic rule is given by

$$Q_k^{\,z}(s)=\begin{cases}\min\{EFT_i(s)\}, & \text{if no conflict exists between jobs}\\ P_{kiji'j'}^{''Z}(tn), & \text{if conflict exists}\end{cases}$$
$$P_{kiji'j'}^{''Z}(tn)=\frac{[\varepsilon_{kiji'j'}(tn)]^{\gamma}.[\psi_i(s)]^{\omega}}{\sum_{(i,j):Oij\in G_k(s)}[\varepsilon_{kiji'j'}(tn)]^{\gamma}.[\psi_i(s)]^{\omega}},\quad \forall(i,j):O_{ij}\in G_k(s) \tag{11}$$

where $\psi_i(s)$ represents a heuristic information and is equal to the sum of the processing time of all unassigned operations of job *i* at step *s*, γ and ω are the scaling parameters that control the relative importance of the pheromone trail and the heuristic information and G_k is the set of contending operations to be loaded on machine *k*. The higher the pheromone intensity $\varepsilon_{kiji'j'}(tn)$ and the higher the heuristic information $\psi_i(s)$, the higher will be the probability that operation O_{ij} may precede other contending jobs on machine *k*. Each ant generates one feasible schedule and the evaluation parameter makespan time is found from the feasible schedule generated by the ant. This procedure is repeated for all ants.
Sorting module: The best solution of the current iteration (*ibest*) and the global best (*gbest*) are sorted and stored separately.
Termination Check module: A specified number of iterations (*no_iter*) is estimated to terminate the algorithm depending on the size of the problem. Termination directs to the output module; otherwise, continue to the pheromone updating module.
Pheromone updating module: At the end of iteration, the pheromone trails corresponding to only one single ant is updated. This ant may be the one which found the best solution in the current iteration (*ibest*) or the one which found the best solution from the beginning of the trial (*gbest*). This pheromone trail update rule is similar to the rule used in Max-Min Ant System proposed by Stützle and Hoos (2000). The pheromone trail update rule corresponding to pheromone trails $\tau_{ijr}(tn)$ and $\varepsilon_{kiji'j'}(tn)$ is given as

$$\tau_{ijr}(tn+1)=\rho.\tau_{ijr}(tn)\quad \forall i,j,r \tag{12}$$

$$\varepsilon_{kiji'j'}(tn+1)=\rho.\varepsilon_{kiji'j'}(tn)\quad \forall i,j,i',j',k \tag{13}$$

$$\tau_{ijr}(tn+1)=\tau_{ijr}(tn+1)+\Delta\tau_{ijr}(best)\quad \forall i,j,r:O_{ij}\ \textit{is assigned to route } r \tag{14}$$

$$\begin{gathered}\varepsilon_{kiji'j'}(tn+1)=\varepsilon_{kiji'j'}(tn+1)+\Delta\varepsilon_{kiji'j'}(best)\\ \forall k,(i,j),(i',j'),s:O_{ij}\in Q_k^{\,z}(s+1),\ O_{i'j'}\in Q_k^{\,z}(s)\end{gathered} \tag{15}$$

$$\Delta\tau_{ijr}(best)=\frac{1}{f(sbest)} \tag{16}$$

$$\Delta\varepsilon_{kiji'j'}(best) = \frac{1}{f(sbest)} \tag{17}$$

where ρ is the evaporation factor in the range [0, 1]. In Equation (16) and Equation (17), *f*(*sbest*) denotes the makespan time of either the iteration best (*ibest*) or the global best solution (*gbest*). When using *gbest* alone, the search concentrates too fast around this solution, thus getting trapped in poor quality solutions and this danger is reduced when *ibest* is chosen for the pheromone trail update as *ibest* solutions may differ from iteration to iteration and a large number of solution components may receive occasional reinforcement. A dynamically mixed strategy of pheromone updating is used where *ibest* is chosen as default for updating the pheromones and using *gbest* only every fixed number of iterations. The frequency of using *gbest* for the pheromone update is increased during the search.
To avoid stagnation of the search the range of possible pheromone trails on each solution component is limited to an interval [$\tau_{max}(tn)$, $\tau_{min}(tn)$] and [$\varepsilon_{max}(tn)$, $\varepsilon_{min}(tn)$] corresponding to pheromone trails $\tau_{ijr}(tn)$ and $\varepsilon_{kiji'j'}(tn)$ respectively. The pheromone trails are deliberately initialized to $\tau_{max}(1)$ and $\varepsilon_{max}(1)$ in order to achieve higher exploration of solutions at the start of the algorithm. In every iteration, it is ensured that pheromone trail respects the limits, i.e., in case of $\tau_{ijr}(tn)>\tau_{max}(tn)$, or $\tau_{ijr}(tn)<\tau_{min}(tn)$, then trail intensities are set to $\tau_{ijr}(tn)=\tau_{max}(tn)$ and $\tau_{ijr}(tn)=\tau_{min}(tn)$, respectively. The values are determined as follows:

$$\tau_{max}(tn+1)=1/(1-\rho).f(gbest) \tag{18}$$

$$\varepsilon_{max}(tn+1)=1/(1-\rho).f(gbest) \tag{19}$$

$$\tau_{min}(tn+1)=\tau_{max}(tn+1)/y \tag{20}$$

$$\varepsilon_{min}(tn+1)=\varepsilon_{max}(tn+1)/y \tag{21}$$

where y is a parameter that defines the space of the region between the limits [$\tau_{max}(tn)$, $\tau_{min}(tn)$] and [$\varepsilon_{max}(tn)$, $\varepsilon_{\min}(tn)$]. Hence, each time a new best solution is found, $\tau_{max}(tn)$ and $\varepsilon_{max}(tn)$ are updated.
Output Module: This module prints the global best solution of the optimal route choices of all operations and schedule for minimum makespan time criterion.

4. Numerical Illustration for the proposed ACO

Table 1 provides the process data of 3 jobs - 5 machines problem that is used for illustrating the proposed ACO.

Job i	Operation j	Number of route choices R_{ij}	Machine No. with Processing time K_{ijr} (T_{ijr}) corresponding to each route r	
			r =1	r =2
1	1	2	2 (3)	3 (7)
	2	2	1 (4)	4 (2)
	3	2	1 (1)	2 (2)
2	1	2	2 (5)	5 (2)
	2	2	2 (3)	3 (6)
	3	2	1 (3)	5 (7)
3	1	2	2 (4)	3 (5)
	2	2	1 (2)	4 (3)
	3	2	1 (2)	3 (3)

Table 1. Process data of the illustration problem

The above data is given as input in the input module. The number of ants used in this problem is 10. The pheromone trails, $\tau_{ijr}(1)$ and $\varepsilon_{kiji'j'}(1)$ are initialized as $\tau_{max}(1)=\varepsilon_{max}(1)=0.1$. In the Ist Stage of solution construction ants reduce the alternate route choice problem into a fixed route problem by allocating operations to the routes using the probabilistic rule given in Equation (10). The parameters used for solution construction are: α=1, β=2. The Ist stage of solution construction for Ant-1 is shown in Table 2. After allocating all the operations to the routes, the ants generate feasible schedules using Giffler and Thompson procedure (1960) in the IInd stage. The active feasible schedule formed by ant z=1 is shown in Table 3. Conflicts resolved during schedule generation are shown in Table 4. The parameters set for the probabilistic rule for schedule generation is: γ=1, ω=2.

Jobs i	Operations j	Route r	Machine k	$\tau_{ijr}(1)$	T_{ijr}	$P'1_{ijr}(1)$	Cumulative $P'1_{ijr}(1)$	$rand()$	Assigned route
1	1	1	2	0.1	3	0.84	0.84	0.041	1
		2	3	0.1	7	0.16	1.0		
	2	1	1	0.1	4	0.2	0.2	0.467	2
		2	4	0.1	2	0.8	1.0		
	3	1	1	0.1	1	0.8	0.8	0.334	1
		2	2	0.1	2	0.2	1.0		
2	1	1	2	0.1	5	0.13	0.13	0.5	2
		2	5	0.1	2	0.87	1.0		
	2	1	2	0.1	3	0.8	0.8	0.169	1
		2	3	0.1	6	0.2	1.0		
	3	1	1	0.1	3	0.84	0.84	0.724	1
		2	5	0.1	7	0.16	1.0		
3	1	1	2	0.1	4	0.61	0.61	0.478	1
		2	3	0.1	5	0.39	1.0		
	2	1	1	0.1	2	0.69	0.69	0.358	1
		2	4	0.1	3	0.31	1.0		
	3	1	1	0.1	1	0.9	0.9	0.962	2
		2	3	0.1	3	0.1	1.0		

Table 2. Solution construction of ant z=1 in stage-I

Machine k	Job i	Steps of schedule generation s								
		1	2	3	4	5	6	7	8	9
1	1								15	15*
	2			8*						
	3					11				
2	1	3	3	8	8	12	12			
	2		5							
	3	4	4	9	9					
3	1									
	2									
	3						14	14	14*	
4	1							14		
	2									
	3									
5	1									
	2	2								
	3									
Datum Time		2	3	8	8	11	12	14	14	15**
Conflict		--	I	--	II	--	--	--	--	--
Sequence of operations $Q_k^z(s)$		O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{11}	O_{12}	O_{33}	O_{13}

*Flow time of jobs **Makespan time

Table 3. Active feasible schedule generation for solution generated by ant z=1

Conflict No.	k	Contending Jobs i	$\varepsilon_{kiji'j'}(1)$	$\psi_i(s)$	$P''1_{kiji'j'}(1)$	Cumulative $P''1_{kiji'j'}(1)$	$rand()$	Assigned Job
I	2	1	0.1	6	0.235	0.235	0.464	2
		2	0.1	6	0.235	0.470		
		3	0.1	9	0.530	1.000		
II	2	1	0.1	6	0.308	0.308	0.705	3
		3	0.1	9	0.692	1.000		

Table 4. Conflict resolved during solution generation by ant z=1

After all the ants have generated a feasible schedule, the makespan time is determined and is shown in Table 5. The best solution found in the current iteration (tn=1) is $f(ibest)$ = 12 corresponding to the solution generated by ant z=2. The current *ibest* solution is compared with the *gbest* solution and if $f(ibest)<f(gbest)$, then the global best solution is updated with the iteration best solution.

Ant z	1	2*	3	4	5	6	7	8	9	10
Makespan time	15	12**	17	13	17	13	21	12	13	13

**ibest* ant **$f(ibest)$ solution

Table 5. Solution generated by all ants in first iteration

The dynamic mixed strategy between *ibest* and *gbest* for pheromone updating is given below:

fq^{gbest} is the number of iterations for which the *ibest* solution is used for pheromone updating. A *gbest* solution is used for pheromone updating after fq^{gbest} no. of iterations. This procedure of pheromone updating is repeated until fq^{gbest} value changes according to iteration number *tn*. In the first 15 iterations (tn<=15), only *ibest* solution is used to update the pheromone trails. The value of fq^{gbest} is set at a higher value of 3 for iterations 15<tn<=30 due to which more number of *ibest* solutions are subjected to pheromone updating. This leads to more exploration of the solution space. The value of fq^{gbest} is set at 2 for 30<tn<=50 and fq^{gbest}=1 for tn>50. The parameters set for pheromone update rule are: ρ=0.9 and y=10. The values of $\tau_{max}(1)$, $\tau_{min}(1)$, $\varepsilon_{max}(1)$ and $\varepsilon_{min}(1)$ are updated and the pheromone intensities are limited to the interval [$\tau_{max}(2)$, $\tau_{min}(2)$] and [$\varepsilon_{max}(2)$, $\varepsilon_{min}(2)$]. The updated values are $\tau_{max}(2)=\varepsilon_{max}(2)=0.833$ and $\tau_{min}(2)=\varepsilon_{min}(2)=0.083$. The pheromone trail intensities for $\tau_{ijr}(tn)$ are updated and is shown in Table 6. The updated pheromone trail intensities for $\varepsilon_{kiji'j'}(2)$ obtained with machines 1 to 5 are shown in Tables 7 to 11 respectively. The process of solution construction, evaluation, sorting and pheromone updating is repeated till the termination criterion is reached.

Route r	Operation O_{ij}								
	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}
1	0.173	0.090	0.090	0.090	0.173	0.090	0.090	0.090	0.173
2	0.090	0.173	0.173	0.173	0.090	0.173	0.173	0.173	0.090

Table 6. Updated pheromone trail intensities for $\tau_{ijr}(2)$

Operation $O_{i'j'}$	Operation O_{ij}				
	O_{12}	O_{13}	O_{23}	O_{32}	O_{33}
0	0.090	0.173	0.090	0.090	0.090
O_{12}	---	0.090	0.090	0.090	0.090
O_{13}	---	---	0.090	0.173	0.090
O_{23}	0.090	0.090	---	0.090	0.090
O_{32}	0.090	0.090	0.090	---	0.090
O_{33}	0.090	0.090	0.090	---	---

Table 7. Updated pheromone trail intensities for $\varepsilon_{kiji'j'}(2)$ for machine k =1

Operation $O_{i'j'}$	Operation O_{ij}				
	O_{11}	O_{13}	O_{21}	O_{22}	O_{31}
0	0.173	0.090	0.090	0.090	0.090
O_{11}	---	0.090	0.090	0.173	0.090
O_{13}	---	---	0.090	0.090	0.090
O_{21}	0.090	0.090	---	0.090	0.090
O_{22}	0.090	0.090	---	0.090	0.090
O_{31}	0.090	0.090	0.090	0.090	---

Table 8. Updated pheromone trail intensities for $\varepsilon_{kiji'j'}(2)$ for machine k =2

Operation $O_{i'j'}$	Operation O_{ij}			
	O_{11}	O_{22}	O_{31}	O_{33}
0	0.090	0.090	0.173	0.090
O_{11}	---	0.090	0.090	0.090
O_{22}	0.090	---	0.090	0.090
O_{31}	0.090	0.090	---	0.173
O_{33}	0.090	0.090	---	---

Table 9. Updated pheromone trail intensities for $\varepsilon_{kiji'j'}(2)$ for machine k =3

Operation $O_{i'j'}$	Operation O_{ij}	
	O_{12}	O_{32}
0	0.173	0.090
O_{12}	---	0.090
O_{32}	0.090	---

Table 10. Updated pheromone trail intensities for $\varepsilon_{kiji'j'}(2)$ for machine k =4

Operation $O_{i'j'}$	Operation O_{ij}	
	O_{21}	O_{23}
0	0.173	0.090
O_{21}	---	0.173
O_{23}	0.090	---

Table 11. Updated pheromone trail intensities for $\varepsilon_{kiji'j'}(2)$ for machine k =5

5. Performance comparison and results

The performance of the proposed ACO algorithm is evaluated by comparing its solutions with the Best Known Solutions (BKS) from literature (Mastrolilli and Gamberdella, 2000). A set of benchmark problems is used for the performance evaluation. The first set of benchmark instances are from Thomalla (2001), in which all the problems are flexible job shop instances with total flexibility, i.e., all the operations in each of the problem instances can be performed on all the machines. The second set of benchmark instances are from Brandimarte (1993), in which all the problems are flexible job shop instances with partial flexibility. The results of the proposed ACO algorithm are evolved with the program coded in C language.

Researchers and practitioners are nowadays using constraint programming (CP) techniques, which is a relatively new methodology for solving combinatorial optimization problems (Bockmayr & Kasper, 1998). Constraint programming can solve an optimization problem by solving a series of decision (or constraint satisfaction) problems resulting from a dichotomy to locate the optimal objective value (Pan & Shi, 2008). More precisely, each problem is solved through an enumerative search that resembles the tree search of branch-and-bound, except that it involves no bounding and only seeks a feasible solution. However, the overall optimum seeking can be computation-intensive. ILOG Solver, a constraint programming tool marketed by ILOG, has proven to solve scheduling problems in reasonable computational time (Heisig & Minner, 1999; Pinedo, 2005). Weil et al. (1995) demonstrated the efficiency of ILOG Solver as a modeling and resolution tool for nurse scheduling problem. Quiroga et al. (2005) developed a CP model for a FMS scheduling problem using ILOG OPL and presented its computational efficiency for benchmark problems. In this chapter, a CP model is developed for the FJSP using ILOG OPL language and solved using ILOG OPL Studio. The results of ILOG OPL Studio are used to test the performance of the proposed ACO for the benchmark instances.

The parameters set for the proposed ACO are: α=1, β=2, γ=1, ω=2 $\varepsilon_{max}(1)=\tau_{max}(1)$=0.1, ρ=0.9 and y=10. In the first 100 iterations, only *ibest* solution is used to update the pheromone trails; fq^{gbest}=4, for 100<tn<=200; fq^{gbest}=3 for 200<tn<=300; fq^{gbest}=2 for 300<tn<=400; fq^{gbest}=1 for 400<tn<=500; fq^{gbest}=0 for tn>500. The number of ants (*no_ant*) used in the proposed ACO

for solution construction is equal to twice the total number of operations. The termination criterion used for ACO is the total number of iterations which is equal to 100 times the total number of operations of all jobs. The parameters for the proposed ACO are obtained by fine tuning through trials. The proposed algorithm is run five times for each problem and the best solution obtained has been taken for comparison. The proposed model in ILOG OPL is run for a pre-specified period of time which is set as the total time required for the proposed ACO to find a solution for a particular problem. Table 12 shows the results of ACO and ILOG OPL Studio that are obtained with a Pentium-IV 2.4GHz processor.
The comparison between the proposed ACO algorithm and the BKS in the literature for the above benchmark problems reveals that the best solution obtained with ACO for seven out of 13 problems is the same as the best known solutions in the literature. For one problem (MK07) the proposed ACO has given a better result than the best known solution. For three problems (MK04, MK05 & MK06) the results of the proposed ACO is closer to the best known solutions. The proposed algorithm has outperformed ILOG OPL Studio for almost all the Benchmark instances. Therefore, it is inferred from the computational results that the proposed ACO provides better performance than ILOG OPL Studio and is competent with the existing methodologies for FJSP.

Reference	Problem Name	Problem Size $n \times m$	BKS	Makespan time		
				Proposed ACO		ILOG OPL Studio
				Test runs	Best	
Thomalla (2001)	EX1	3×3	117	117,117,117,117,117	117	117
	EX2	4×3	109	109,109,109,109,109	109	109
	EX3	6×10	316	316,316,316,316,316	316	674
Brandimarte (1993)	MK01	10×6	40	40,40,40,40,40	40	52
	MK02	10×6	26	27,27,27,26,27	26	49
	MK03	15×8	204	204,204,204,204,204	204	319
	MK04	15×8	60	67,66,67,67,67	66	67
	MK05	15×4	173	178,178,174,174,178	174	293
	MK06	10×15	58	77,77,80,77,81	77	230
	MK07	20×5	144	143,144,144,144,144	143	223
	MK08	20×10	523	523,523,523,523,523	523	595
	MK09	20×10	307	328,349,346,340,343	328	534
	MK10	20×15	198	247,258,267,248,254	247	385

Table 12. Result obtained with the proposed ACO and ILOG OPL Studio for the set of data from literature

6. Conclusion

In this Chapter, we proposed an ACO based heuristic to solve the FJSP for minimum makespan time criterion. The solution construction method used in the proposed ACO makes it capable to rummage through the entire solution space and provide all possible instances that an enumerative search can and is therefore capable of finding the optimal or near-optimal solutions. Since the proposed ACO uses Giffler and Thompson schedule generation procedure for generating active feasible schedules, therefore, the proposed ACO can be easily adapted to generate schedules for any scheduling objective of FJSP. The performance of the proposed ACO is analyzed with various benchmark instances, which reveals that the proposed ACO is competent with the existing approaches. The proposed ACO has outperformed the CP model solved using ILOG OPL Studio. A future research issue would be to develop hybrid heuristics by incorporating local search techniques such as Tabu search, Simulated Annealing, etc. to the proposed ACO algorithm.

7. References

Baker KR (1974). *Introduction to Sequencing and Scheduling*. Wiley, New York.

Blum, C. (2002). ACO applied to group shop scheduling: A case study on intensification and diversification. In *ANTS 2002*, Dorigo M. et al. (ed.), pp. 14-17, Springer-Verlag, Berlin, Germany.

Blum, C. & Sampels, M. (2004). An ant colony optimization algorithm for shop scheduling problems. *Journal of mathematical modeling and algorithms*. Vol. 3, No. 3, pp. 285-308.

Bockmayr, A. & Kasper, T. (1998). Branch and infer: A unifying framework for integer and finite domain constraint programming. *INFORMS Journal of Computing*, Vol. 10, No. 3, pp.287-300.

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu Search. *Annals of Operations research*, Vol. 41, No. 1, pp.157-183.

Brucker, P. (1995). *Scheduling algorithms*. Springer-Verlag, Berlin-Heidelberg.

Brucker, P. & Schlie, R. (1990). Job shop scheduling with multi-purpose machines. *Computing*, Vol.45, pp.369-375.

Chen, H.; Ihlow, J. & Lehmann, C. (1999). A genetic algorithm for flexible job-shop scheduling. *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 2, pp.1120-1125, IEEE.

Choi, I.C. & Choi, D.S. (2002). A local search algorithm for job shop scheduling problems with alternative operations and sequence-dependent setups. *Computers and Industrial Engineering*, Vol. 42, pp. 43-58.

Chryssolouris, G. & Chan, S. (1985). An integrated approach to process planning and scheduling. *Annals of CIRP*, Vol. 34, pp.413-417.

Colorni, A.; Dorigo, M.; Maniezzo, V. & Trubian, M. (1994). Ant System for Job Shop Scheduling**.** *JORBEL – Belgian Journal of Operations Research, Statistics and computer science,* Vol. 34, No.1, pp.39-53.

Conway, R.W.; Maxwell, W.L. & Miller, L.M. (1967). *Theory of Scheduling,* Addison-Wesley Reading, MA.

Dauzere-Peres, S. & Paulli, J. (1997). An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem with tabu search. *Annals of Operations Research*, Vol. 70, pp.281-306.

den Besten M.; Stützle T. & Dorigo, M. (2000). An ant colony optimization application to the single machine total weighted tardiness problem. In: *Proc. of ANTS'2000*, Dorigo, M.; Middendorf, M. & Stützle, T. (eds), pp. 39–42, Belgium.

Dorigo, M.; Colorni, A. & Maniezzo, V. (1992). An investigation of some properties of an ant algorithm. In: *Proceedings of the conference on parallel problem solving from nature,* R. Manner & B. Manderick (eds), pp 509–520, Elsevier, Brussels, Amsterdam.

Dorigo, M. & Stutzle, T. (1999). *Ant colony algorithms for quadratic assignment problem: new ideas in optimisation*. McGraw Hill, New York.

Fattahi, P.; Mehrabad, M.S. & Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, Vol. 18, No. 3, pp. 331-342.

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

Gajpal, Y. & Rajendran, C. (2006). An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshop. *International Journal of Production Economics*, Vol. 101, No. 2, pp.259-272.

Garey, M.R.; Johnson, D.S. & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, Vol. 1, pp.117-129.

Giffler, B. & Thompson, G.L. (1960). Algorithms for solving production scheduling problems. *Operations Research*, Vol. 8, pp.487-503.

Girish, B.S. & Jawahar, N. (2008). Scheduling job shops associated with multiple routings with genetic and ant colony heuristics. *International journal of production research*, In print, available online DOI:10.1080/00207540701824845.

Groover, M.P. (2003). *Automation, production systems, and computer integrated manufacturing*. Prentice Hall of India Pvt Ltd, New Delhi, India.

Hankins, S.L.; Wysk, R.A. & Fox, K.R. (1984). Using a CATS database for alternative machine loading. *Journal of Manufacturing Systems*, Vol.3, pp.115-120.

Heinonen, J. & Pettersson, F. (2007). Job-shop scheduling and visibility studies with a hybrid ACO algorithm. In:, *Swarm intelligence: Focus on ant and particle swarm optimization,* Chan, F. T. S. and Tiwari, M. K. (eds.), pp. 355 – 372, Itech Education and Publishing, Vienna, Austria.

Heisig, G. & Minner, S. (1999). ILOG OPL Studio. *OR Spectrum*, Vol. 21, No. 4, pp.419-427.

Ho, N.B. & Tay, J.C. (2004). GENACE: An Efficient Cultural Algorithm for Solving the Flexible Job-Shop Problem. *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol.1, pp.1759-1766, IEEE.

Ho, N.B.; Tay, J.C. & Lai, E.M.K. (2007). An effective architecture for learning and evolving flexible job-shop schedules. *European Journal of Operational Research*, Vol. 179, pp.316-333.

Hoitomt, D.J.; Luh, P.B. & Pattipati, K.R. (1993). A practical approach to job shop scheduling problems. *IEEE transactions on Robotics and Automation*, Vol. 9, No.1, pp.1-13.

Huang, K. & Liao, C. (2008). Ant colony optimization combined with taboo search for the job shop scheduling problem. *Computers and Operations Research*, Vol. 35, No. 4, pp.1030-1046.

Hurink, J., Jurisch, B. & Thole, M. (1994). Tabu search for the job-shop scheduling with multi-purpose machines. *OR Spektrum*, Vol. 15, pp. 205-215.

Hussain, M.F. and Joshi, S.B. (1998). A Genetic Algorithm for Job Shop Scheduling problems with Alternate Routing. *Proceedings of IEEE International conference on systems, man and cybernetics*, Vol.3, pp. 2225-2230.

Iwata, K.; Murotsu, Y.; Oba, F. & Okamura, K. (1980). Solution of large-scale scheduling problems for job-shop type machining systems with alternative machine tools. *Annals of the CIRP*, Vol. 29, pp. 335-338.

Iwata, K.; Murotsu, Y.; Oba, F. & Uemura, T. (1978). Optimization of selection of machine tools, loading sequence of parts and machining conditions in job-shop type machining systems. *Annals of the CIRP*, Vol.27, pp.447-451.

Jain, A.S. & Meeran, S. (1999). Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, Vol.113, No.2, pp. 390-434.

Kutanoglu, E. & Sabuncuoglu, I. (1999). An analysis of heuristics in a dynamic job shop with weighted tardiness objectives. *International Journal of Production Research*, Vol. 37, No.1, pp.165-187.

Liao, C. & Juan, H. (2007). An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups. *Computers and operations research*, Vol. 34, No. 7, pp.1899-1909.

Mastrolilli, M. & Gamberdella, L.M. (2000). Effective neighbourhood for the flexible job shop problem. *Journal of Scheduling*, Vol. 3, No.1, pp.3-20.

Mehrabad, M.S. & Fattahi, P. (2007). Flexible job shop scheduling with tabu search algorithms. *The International Journal of Advanced Manufacturing Technology*, Vol. 32, No. 5-6, pp. 563-570.

Merkle, D.; Middendorf, M. & Schmeck, H. (2002). Ant Colony Optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary computation*, Vol. 6, No. 4, pp.333-345.

Mesghouni, K.; Hammadi, S. & Borne, P. (1998). Evolution programs for job shop scheduling. *Proceedings of the IEEE international conference on computational cybernetics and simulation*, Vol. 1, pp.720-725.

Moon, J. & Lee. J. (2000) Genetic Algorithm Application to the Job Shop Scheduling problem with Alternative Routing. *Technical report-Brain Korea 21 logistics Team*, Pusan National University.

Najid, N.M., Dauzere-Peres, S. & Zaidat, A. (2002). A modified simulated annealing method for flexible job shop scheduling problem. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Vol. 5, pp. 6-9.

Nasr, N. & Elsayed, E.A. (1990). Job Shop Scheduling with Alternative Machines. *International Journal of Production Research*, Vol. 28, pp. 1595-1609.

Pan, Y. & Shi, L. (2008). New hybrid optimization algorithms for machine scheduling problems. *IEEE Transactions on Automation Science and Engineering*, Vol. 5, No. 2.

Pinedo, M.L. (1995). *Scheduling: theory, algorithms and systems*. Englewoodcliffs, New Jersey.

Pinedo, M.L. (2005). *Planning and scheduling in manufacturing and services*. Springer, New York.

Quiroga, O.; Zeballos, L. & Henning G.A. (2005). Constraint Programming approach to tool allocation and resource scheduling in FMS. In: *Proceedings of the 2005 IEEE International conference on Robotics and automation ICRA-2005*, pp. 3715-3720.

Rajendran, C. & Ziegler, H. (2004). Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research*, Vol. 155, No. 2, pp.426–438.

Rossi, A. & Dini, G. (2007). Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimization method. *Robotics and Computer-Integrated Manufacturing*, Vol. 23, No. 5, pp.503-516.

Stutzle, T. & Hoos, H.H. (2000). Max-min ant system. *Future Generation Computer Systems*, Vol. 16, pp.889–914.

Tay, J.C. & Ho, N.B. (2008). Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers and Industrial Engineering*, Vol. 54, pp.453-473.

Thomalla, C.S. (2001). Job shop scheduling with alternative process plans. *International Journal of Production Economics*, Vol.74, No.1-3, pp.125-134.

Vairaktarakis, G.L. & Cai, X. (2003). The value of processing flexibility in multipurpose machines. *IIE transactions*, Vol. 35, pp.763-774.

Weil, G.; Heus K.; Francois, P. & Poujade, M. (1995). Constraint programming for nurse scheduling. *IEEE Engineering in medicine and biology magazine*, Vol. 14, No. 4, pp.417-422.

Wilhelm, W. & Shin, H. (1985). Effectiveness of alternative operations in a flexible manufacturing system. *International Journal of Production Research*, Vol. 23, pp.65-79.