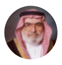
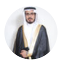
# Ant Colony Optimization for Job Shop Scheduling to Minimize the Total Weighted Tardiness





*Proceedings of the 2014 Industrial and Systems Engineering Research Conference*
*Y. Guan and H. Liao, eds.*

# Ant Colony Optimization for Job Shop Scheduling to Minimize the Total Weighted Tardiness

**Mustufa H. Abidi[a], Ibrahim Al-Harkan[b], Abdulaziz M. El-Tamimi[b], A.M. Al-Ahmari[a,b], Emad S. Abouel Nasr[b,c]**

a. Advanced Manufacturing Institute, College of Engineering,
King Saud University, Riyadh-11421, KSA

b. Industrial Engineering Department, College of Engineering,
King Saud University, Riyadh-11421, KSA

c. Department of Mechanical Engineering, Faculty of Engineering
Helwan University, Cairo, Egypt

## Abstract

In modern manufacturing systems, due date related performance is becoming increasingly important in maintaining a high service reputation. However, compared with the extensive research on make span minimization, research on the total weighted tardiness objective is rather meager, partly because this objective function is more difficult and complex to optimize. In this research work, focus is on the job shop scheduling problem with the objective of minimizing total weighted tardiness. Ant Colony Optimization is a good technique for many combinatorial types of optimization problems like the Travelling Salesman Problem. A combined Shifting Bottleneck and Ant Colony Optimization technique to solve the Job Shop Problem to reduce the total weighted tardiness is implemented successfully. The Shifting Bottleneck heuristic is used to generate an initial solution and then the ant colony optimization is applied for further minimizing the total weighted tardiness in job shops. A Matlab program is developed based on this combined algorithm and it is tested on the standard benchmark sets and compared with the other approaches. The implemented heuristic (named SB-ACO) shows better results when compared with other heuristics such as pure Simulated Annealing, pure Tabu Search and others.

## Keywords
Job Shop, Shifting Bottleneck, Ant Colony Optimization (ACO), Simulated Annealing (SA), Weighted Tardiness

## 1. Introduction

Scheduling is the process of allocating tasks, in an efficient and organized manner, to a set of resources. Generally, these tasks require the use of varied resources while the resources, on the other hand, are limited in nature with respect to quantity as well as the time that they are available for. In addition, the tasks may be required to be completed within a certain time and in a certain sequence. These dynamics imply that tasks as well as resources have different constraints, thus making scheduling a complicated process.
The practice of scheduling is very common in the real world; the most common among them being the scheduling of tasks in a manufacturing plant, school timetable scheduling and airplane scheduling. This work will address the problems of scheduling in a Job Shop manufacturing environment; hence, the next few sections focus on this area, the problems faced therein, past approaches and the methodology to solve the problem.
Manufacturing scheduling refers to the scheduling of jobs on machines so that they may be processed in an efficient and cost effective manner. The process of scheduling requires identifying some characteristics of the manufacturing system like the type of layout and the shop configuration. Shop configuration refers to the organization of the flow of products in the production unit. The way in which the jobs are scheduled will depend on the type of shop configuration. The different types of shop configuration are single machine shop, parallel machine shop, flow shop,

job shop, assembly lines, and manufacturing cells. The job shop scheduling problem (JSSP) has been an important research focus in both theoretical and industrial fields ever since the 1950s. Concerning its difficulty, JSSP was shown to be *NP*-hard in the strong sense [1]. It is very hard to find the optimal solution even for small JSSP instances. The well-known JSSP benchmark problem “MT10” was not solved for optimality until two decades after its creation [2].

In the last three decades, the problem has captured the attention of many researchers and many research methodologies have been proposed such as integer programming, approximation algorithms, meta-heuristics etc. For smaller sets of problems, the optimization algorithms based on branch & bound, linear programming, and Lagrangian relaxation have given the optimal and good solution, but as the problem size increases, it is very complex and difficult to handle such algorithms resulting into very high computational time. Therefore a lot of interest has been shown by researchers in meta-heuristic and local search optimization strategies for solving JSSP, such as simulated annealing (SA) [3, 4 ,5], genetic algorithm (GA) [6,7], Tabu search (TS) [8], scatter search (SS) [9,10], particle swarm optimization (PSO) [11,12] and ant colony optimization (ACO) [13,14]. In the last five to seven years, researchers have paid a lot of attention to hybrid algorithms to solve the JSSP. They are very efficient algorithms because they use the advantage of two or more algorithms together. The Initial solution plays an important role in scheduling problems, therefore, most of the time, one technique is used to find the initial solution and then the other is applied to find the optimal or near to optimal solution. It has been verified by experiments that these methods can find high-quality solutions within reasonable computational time. These have made the research on intelligent hybrid algorithm for JSSP increasingly popular in the recent years. Therefore, in this paper, a hybrid technique is applied in which the initial solution is generated with shifting bottleneck (SB) technique, and then it is feed to the ant colony optimization algorithm to reach to the optimal or near to optimal solution. This hybrid technique is applied to minimize the total weighted tardiness in the job shop scheduling.

The rest of this paper is organized as follows. In Section 2, we will present a literature survey for JSSP and applied algorithms. In Section 3, we will illustrate the proposed hybrid algorithm for finding a good enough schedule from the search space. In Section 4, we will present the test results by applying the algorithm to the standard 10 x 10 JSSP, and demonstrate the quality of the solution obtained by the proposed algorithm in comparison with those obtained by the other techniques. Finally, we will make a conclusion in Section 5.

## 2. Literature Survey

The JSSP has been studied by many authors and it is considered NP-hard. Several methods have been proposed by authors to solve this problem as discussed in the previous section. Among the approaches that have obtained good results are: effective branch and bound methods developed by Carlier & Pison in 1989 in which he reached optimal solutions for instances 10×10 (first refers to number of jobs and second to number of machines) [15]. Effective shifting bottleneck heuristic developed by Adams et al. in 1988 also showed good results in instances of different sizes [2]. For problems bigger than 10x10 local search such as Tabu search, Simulated annealing was accomplished by Van Laarhoven et al. in 1992 [16]. Davis in 1985 [17] was the first person to apply Genetic algorithm to the Job Shop problem.

Now, there are many such works, e.g., a genetic algorithm with search area adaptation was proposed by Someya and Yamamura [18]. Zhou and Feng [19] proposed a hybrid heuristics GA for JSSP, where the scheduling rules, such as shortest processing time and most work remaining, were integrated into the process of genetic evolution. Park et al. developed an efficient method based on the genetic algorithm to address JSSP. A scheduling method based on a parallel genetic algorithm was designed [20]. However, present genetic algorithms for JSSP are generally with a slow convergence speed and are easy to trap into local optimal solutions. In order to augment the convergence speed, many researchers concentrated on combining genetic algorithm with local search methods to develop some hybrid optimization strategies for JSSP. Wang and Zheng [21] combined simulated annealing algorithm and genetic algorithm to solve JSSP. Goncalves et al. [22] presented a hybrid genetic algorithm for JSSP, in which the schedules were constructed by using a priority rule and the priorities were defined by the genetic algorithm, and then a local search heuristic was applied to improve the solution. Zhang proposed a G-SA to solve the JSSP by combining the GA and SA [23]. Zhang solved the JSSP by combining the Tabu Search (TS) and the GA [24]. However, there are still many shortcomings in these algorithms, and these are pointed out by Qing-dao-er-ji and Wang [25].

### 2.1 Job shop scheduling problem to minimize weighted tardiness

Asano et al. [26], considers the job shop scheduling problem to minimize the total weighted tardiness with job-specific due dates and delay penalties, and a heuristic algorithm based on tree search procedure was developed for solving the problem. Computational results for benchmark problems of 10 x 10 and 15 x 15 showed that the proposed algorithm could find the sub-optimal solutions but in less computation time.

Zhou et al. [27], developed a hybrid framework integrating a heuristic and a genetic algorithm for job-shop scheduling to minimize weighted tardiness. The results showed that the hybrid framework performed significantly better than does either a heuristic or GA alone.
Bulbul K. [28], studied the job shop scheduling problem with the objective of minimizing the total weighted tardiness. He proposed a hybrid shifting bottleneck-tabu search (SB-TS) algorithm by replacing the re-optimization step in the shifting bottleneck (SB) algorithm by a tabu search (TS). Singer and Piedo [29], presented and compared a number of branch and bound algorithms for minimizing the total weighted tardiness in job shops. They obtained optimal solutions for all the instances with ten jobs and ten machines that were considered, but the procedure required high computational power and time.

### 2.2 Job shop scheduling problem with Ant Colony Optimization

Bauer et al. [30], considered the Single Machine Total Tardiness Problem. To solve this NP-hard problem, they applied the ant colony optimization metaphor. They test their algorithm using 125 benchmark problems and present computational results. Colorni et al. [31], showed that how a new heuristic called *Ant System*, in which the search task is distributed over many simple, loosely interacting agents can be successfully applied to find good solutions of job-shop scheduling problems. The objective function is to minimize the maximum makespan.
Zwaan et al. [32], introduced and used the Ant System to solve the problem of job shop scheduling. In this paper, the authors presented some statistical analysis for parameter tuning and compared the quality of obtained solutions for well-known benchmark problems in job shop scheduling. Puris et al. [33], introduced a multilevel approach of Ant Colony Optimization to solve the Job Shop Scheduling Problem (JSSP). Experimental results obtained conclude that the Two- Stage approach significantly reduces the computational time to get a solution similar to the Ant System.
Raghavan et al. [34], considered a scheduling problem in a system of parallel processors with the objective of minimizing total weighted tardiness. An extensive experimentation was conducted to evaluate the performance of the ACO approach on different problem sizes with the varied tardiness factors. The experimentation showed that the proposed ant colony optimization algorithm is giving promising results compared to the best of the available heuristics.
Besten et al. [35], presented an application of the Ant Colony Optimization (ACO) metaheuristic to the single machine total weighted tardiness problem. First, a brief discussion about the constructive phase of ACO in which a colony of artificial ants generates a set of feasible solutions was given. Then, the authors introduced some simple but very effective local search. At last, they combined the constructive phase with local search to obtain a novel ACO algorithm that uses a heterogeneous colony of ants and is highly effective in finding the best-known solutions on all instances of a widely used set of benchmark problems.
According to best of author's knowledge from the literature survey, it can be said that there has been no work done for Job Shop Scheduling using the Ant Colony Optimization (ACO) Technique to minimize the total weighted tardiness. Therefore, authors want to use this technique for this type of problem. However, from the literature also we have read that ACO takes more time to converge if it is away from the optimal solution so we use the shifting bottleneck heuristic to find the initial solution and then feed that to the ACO.

## 3. Problem Definition and Methodology

JSSP is a special type of scheduling problem; the algorithm constructed in this paper attempts to solve this specific problem. The job shop problem pertains to the allocation of jobs to machines in a job shop environment. A Job Shop is a production center where all of the machines possessed by the center are placed in the same area and all jobs entering the center share the machines [5]. The general representation of the job shop scheduling problem with the objective to minimize the total weighted tardiness is shown in equation (1):

$$J_m || \sum w_j T_j \tag{1}$$

Where $J_m$: Machine environment i.e. Job Shop here.
$w_j$: Weight assigned to Job j.
$T_j$: Tardiness of Job j; max. $\{C_j - d_j, 0\}$
$C_j$: Completion Time of Job j.
$d_j$: due date of Job j.

*Therefore, the objective of this study is to minimize the total weighted tardiness in a job shop.*

As discussed in the first chapter, the Job shop problem pertains to the allocation of jobs to machines in a job shop environment. A Job Shop is a production center where all of the machines possessed by the center are placed in the same area and all jobs entering the center share the machines. It is possible that a particular job entering the system will not need all of the machines that are in the center and it is also possible to have multiple quantities of the same type of machine. This makes the process of scheduling all the more difficult as all of the jobs that enter the system will go to most of the machines, and therefore, there is a high probability of jobs waiting for busy machines or vice versa if improper scheduling procedures are followed, resulting in poor productivity within the production system. Figure 1 gives an idea of what a Job Shop looks like and how jobs entering the system move between the machines in their respective processing orders for specified processing time periods.

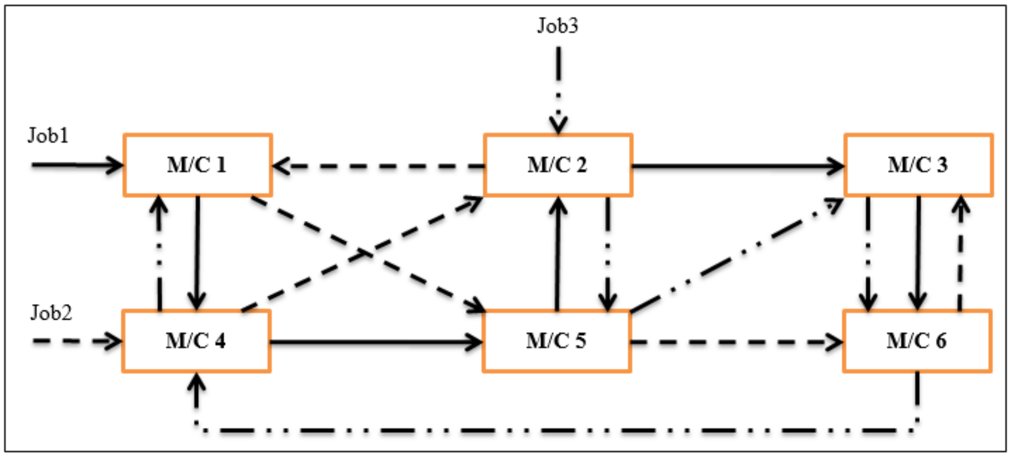


Figure 1: A Job Shop with 6 machines and 3 Jobs

The complications involved in any job shop problem would address the following issues:

- Reduction of the possibility of assigning 2 jobs to the same machine at the same time.
- In the case of conflict above, which job gets the machine first and which job waits for the machine.
- Final aim, in our case, is to reduce the sum of the weighted tardiness of the jobs.

## 3.1 Problem Definition and Assumptions

A job shop scheduling problem is solved with an objective of minimizing the total weighted tardiness. We have selected the standard benchmark problems of size 10 x 10 (i.e. 10 jobs and 10 machines problems). We have found that there are 18 problems of such size in the problem database such as MT10, ABZ5, ABZ6, LA16 etc. The problem instances can be found on the OR-Library (http://mscmga.ms.ic.ac.uk). The type of job shop problem considered in this work is a particular type of the general problem that has a few unique characteristics:

- It is assumed that all of the jobs in this problem have to be processed using all of the machines. Hence, if there are 10 machines and 4 jobs, each of the jobs will be processed on all 10 machines and there will be total 40 operations.
- There are no constraints on the processing times of each job on each machine.
- All jobs arrive at the same time and that time (t) is assumed to be t = 0.
- The only constraint in this problem is that of the precedence order of the operations of each job. A job has to follow the order of operations assigned to it and cannot violate that one constraint.
- Each job follows a unique predetermined fixed routing. No recirculation is allowed.
- Preemptions are not allowed.
- Processing times and due dates are known and deterministic.
- There is no machines breakdowns and they are always available during the scheduling.
- The transportation time between different machines and the setup time between different jobs can be neglected.
- Each machine can process at most one job at a time.
- Each job may be processed by at most one machine at a time.

In this research work, a hybrid of shifting bottleneck algorithm and ant colony optimization (SB-ACO) is applied to minimize the weighted tardiness in the job shop scheduling. Shifting bottleneck is applied to obtain the initial solution that is feed to the ACO to find the optimal or near to optimal solution.

### 3.2 Shifting Bottleneck Heuristic

The Shifting Bottleneck Heuristic is a procedure intended to minimize the time it takes to do work, or specifically, the makespan in a job shop. Assuming that the jobs are actually competing for the same resources (machines) then there will always be one or more resources that act as a 'bottleneck' in the processing. This heuristic or 'rule of thumb' procedure minimizes the effect of the bottleneck. The Shifting Bottleneck Heuristic is intended for job shops with a finite number of jobs and a finite number of machines. Bottlenecks are generally recognized as some resources or utilities, which heavily limit the performances of a production system. For different application demands and different operation manners, numerous definitions of what contributes to a bottleneck can be found in the literature.

The Shifting Bottleneck Heuristic is used in manufacturing and service industries that include job shops with constraints on the order that the machines must be used for each job. A precedence constraint in this context is when one machine must be used before another machine on any given job (or patient). These types of problems with multiple machines are known to be computationally very difficult. The processing time of each job on each machine is given. Job *j* being performed on machine *i* is denoted *ij*. It is assumed that each machine can only work on one job at a time. The objective is to determine the schedule that will produce the shortest makespan.

**General Procedure:**

1. Make Graph
    - ✓ Determine starting Tardiness
2. Determine optimal sequence for bottleneck machine (Considering precedence constraints)
3. Perform an iteration
    - ✓ Lowest maximum lateness
    - ✓ Branch and bound technique
    - ✓ Include optimal sequence in graph
4. Determine optimal sequences for remaining machines (Considering precedence and machine constraints).
5. Perform further iterations
    - ✓ Conduct iterations until all machines have been accounted for
    - ✓ Draw out final graph and determine final tardiness.

**Shifting Bottlenecks for Weighted Tardiness**

A shifted bottleneck procedure to a job shop with total weighted tardiness objective:

- ✓ Need *n* sinks in disjunctive graph
- ✓ Machines scheduled one at a time
- ✓ Given current graph calculate completion time of each job
- ✓ Some *n* due dates for each operation and then find Piecewise linear penalty function

**Machine Criticality:**

- ✓ Solve as a single machine problem; $1 \mid \mid h_j(T_j)$
- ✓ May have delayed precedence constraints.
- ✓ Generalizes single-machine with n jobs, precedence constraints, and total weighted tardiness objective.
- ✓ Apply, Apparent Tardiness Cost (ATC) rule. This ATC heuristic is a composite dispatching rule that combines the WSPT rule and the so-called Minimum Slack first (MS) rule (under the MS rule the slack of job j at time t, max($d_j$–$p_j$–t, 0), is computed and the job with the minimum slack is scheduled).

**ATC Rule:**

- ✓ Under the ATC rule jobs are scheduled one at a time; i.e., every time the machine becomes free a ranking index is computed for each remaining job. The job with the highest ranking index is then selected to be processed next.
- ✓ The index is given as:

$$I_j(t) = \frac{w_j}{p_j} \exp\left(-\frac{\max(d_j - p_j - t, 0)}{K\bar{p}}\right)$$

- ✓ Where K is a scaling parameter and p-bar is average of processing time of the remaining jobs.

### 3.3 Ant Colony Optimization

The Ant colony was developed by Marco Dorigo in 1991 [31]. Ant-based systems are population-based stochastic search methods. Ant algorithms are basically a colony of artificial ants or cooperative agents, designed to solve a particular problem. They are a class of heuristics based search algorithms used to solve many combinatorial optimization problems. These algorithms are probabilistic in nature because they avoid the local minima entrapment and provide very good solutions close to the natural solution. They are inspired by the co-operative behavior exhibited by real ants in performing various day-to-day activities such as brooding and foraging. Individual real ants are incapable of performing a structured task but an ant colony has great potential to carry out a coordinative activity. The ants' main medium of communication is through building up of a path through an artificial chemical substance called '*pheromone*'. This method of indirect communication is referred to as 'stigmergy'. Whenever an ant leaves their nest to search for food, they lay a trail of pheromone on their path. The number of ants that have traveled on the path determines strength of the pheromone trail. The ant that travels the shortest path reinforces the path with more amount of pheromone, which aids others to follow. After an initial randomization, the ants finally arrive at a shortest path. This behavior is known as 'auto catalytic' behavior or the positive feedback mechanism in which reinforcement of the previously most followed route, is more desirable for future search [31].

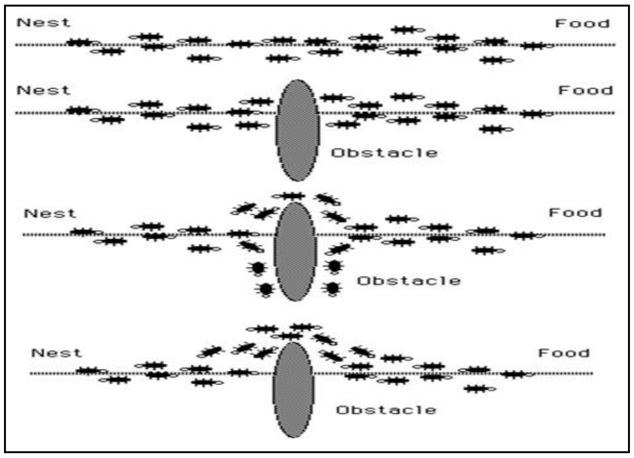


Figure 2: Ants Behavior

Inspired by this probabilistic behavior of real ants, ant algorithms are the software agents that coordinate by updating the information of a common memory similar to the pheromone trail of the real ants. When a large number of these simple artificial agents coordinate based on the memory updation they are able to build good solutions to hard combinatorial optimization problems. Similar to the pheromones, which had a rate of evaporation, the unwanted or poor quality solution is removed from the solution space.

In ACO, the problem is defined in the form of a network or a graph. All possible links between the components of the network and limiting criteria's are identified. The distance between the nodes and an initial amount of pheromone deposition on the nodes are given as the input to the system. Based on the given constraints the possible feasible solutions are identified and a potential search list is created. When the artificial ant agents travel from one node to another the memory is updated based on the goodness of the solution. The iteration continues until the best solution is obtained based on the terminating conditions specified. In most cases, the minimization of the objective function is the goal and the terminating condition is either the number of iterations or the computation time of the problem.

ACO metaheuristic is used to solve the complex NP-hard problem of job-shop. In the scheduling problem, it is essential to find the processing sequence for a sequence of orders where set-up times are sequence dependent. In this work, the basic idea is based on the renowned traveling salesman problem (TSP). Each order to be processed is represented by a node in a network. A matrix *A* shows the distance between each pair (*ij*) of nodes where (*dij*) denotes the setup and processing time needed to complete job *j* if it is preceded by job *i*. When an ant travels from node *i* to node *j*, it will leave a *trail* comparable to the pheromone on the edge (*ij*). The trail stores info associated to the previous use of edge (*ij*) and how frequent this path has been used, the greater the pheromone the greater the probability of choosing it once again. At time *t*, the ant chooses the next node using a probabilistic *visibility* rule. This is a greedy rule preferring the nearer nodes, which in fact are the jobs that are shorter in terms of both setup and processing. The choice probability is also affected by $\tau_{ij}(t)$, the *trail intensity* on edge *(ij)*. Initially, the trail on each edge is set to an arbitrary but small positive level, $\tau_0$. Parameters $\alpha$ and $\beta$ are used to vary the relative significance of the visibility and the trail intensity. To confirm the generation of a realistic tour, nodes that have previously been visited on the current trip are omitted from the choice with the use of tabu list. Each ant has its own tabu list. Various steps for the ACO metaheuristic algorithm are given below:

**STEP –I (Input)**

Number of jobs, Number of machines, Number of operations of each job, processing time of each operation, initial schedule on each machine, f (best).

**STEP –II (Initialization)**

Set ACO parameters, $\beta$, $\tau_0$, $\rho_{local}$, $\rho_{global}$, Q, ITER, ANTS. Initialize $\tau_{ij} = \tau$ , $i \varepsilon I$ and $j \varepsilon J$ . Set iter = 1 and a=1.

**STEP –III (Solution Construction)**

**Step (1):** Generate $q_m \sim U[0,1]$.

**Step (2):** Select the Job $j$ for the selected machine $i$ by the following transition rule:

$$P_{ij} = \begin{cases} \dfrac{\eta_{ij}[\tau_{ij}]^{\beta}}{\sum_{l\epsilon J}\eta_{il}\,[\tau_{ij}]^{\beta}} & if\ eth\ ant\ uses\ edge\ (i,j) in\ its\ sequence\ of\ operations \\ 0\ Otherwise & \end{cases}$$

**Step (3):** Local Pheromone Updation

Update the pheromone levels for the selected machine $i$ and job j as:

$$\tau_{ij} = (1 - \rho_{local})\tau_{ij} + \rho_{local}\,\tau_o$$

**Step (4):** Find the objective function value of the current solution.

if f < f(best), then f(best) = f.

**Step (5):** If a<ANTS, go to STEP-III, otherwise go to step (6).

**Step (6):** Global Pheromone Updation

Update the pheromone levels of the best solution found so far as:

$$\tau_{ij} = (1 - \rho_{global})\tau_{ij} + \rho_{global}\,\Delta\tau_{ij}$$
$$where,\ \Delta\tau_{ij} = \frac{Q}{f(best)}$$

Go to **STEP-IV.**

**STEP –IV (Termination)** if iter $\leq$ ITER go to **STEP-III** otherwise f(best) is the final solution and **STOP.**

**Notations:**

| | |
|---|---|
| n = Number of jobs | $P_j$ = Processing time of job j |
| j = Index for jobs, $\forall j \in J$ | i = Index for machine, $\forall i \in I$ |
| $w_j$ = Weightage or priority of job j | $d_j$ = Due date of job j |
| $C_j$ = Completion time of job j | $T_j$ = Tardiness of job j = max {0,Cj–dj} |
| m = Number of Machines | $t_i$ = Makespan of the scheduled jobs on machine I |
| I = Set of machines ={1,...,m} | J = Set of jobs = {1,...,n} |
| $\eta_i$ = Heuristic information on the machines | $\eta_{ij}$ = Heuristic information on the machine- job pair (i, j) |
| $\beta$ = Relative importance of $\tau_{ij}$ over $\eta_{ij}$ | $\tau_{ij}$ = Pheromone information on the machine- job pair (i, j) |
| $\rho_{local}$ = Local pheromone evaporation rate | $\rho_{global}$ = Global pheromone evaporation rate |
| ANTS = The number of ants in each ant colony | ITER = Total number of iterations to be run |

## 4. Results

As discussed in previous section, 18 standard benchmark problems of size 10x10 for job shop scheduling were chosen.

### 4.1 Initial Solution

We find the Initial Solution using the Shifting Bottleneck technique with the help of **LEKIN®**. It is a Scheduling System developed by Leonard N. Stern School of Business, New York University.

The weights and due dates are assigned to each job as in Singer and Pinedo [29]. According to Singer & Pinedo, usually 20% of customers are very important, 60% are of average importance and the remaining 20% are of less importance. Hence for weighted tardiness problems, 20% of jobs are assigned a weight of 4; 60% are assigned a weight of 2 and the remaining are assigned a weight of 1.

There is a due date tightness factor (*f*):

*f = 1.3 (High Tightness)*; ***f = 1.5 (Medium Tightness)*** and *f = 1.6 (Low Tightness)*

Therefore due date ($d_j$)= $r_j + (f * \sum P_i)$ ; where $r_j$ = release date of jobs

$\sum P_i$ = Sum of processing Time of Job j on all machines.

We have selected the medium tightness factor for our problems and also according to our assumption the release time is equal to zero, therefore we can calculate the due date.

The problem data is input into the Lekin system, which gives the results based on the Shifting bottleneck technique. The minimized weighted tardiness i.e. the objective function based on SB and the final schedule is shown in table 1.

Table 1: Initial Solution schedule for MT10

| **Machines** | **Job** | **Job** | **Job** | **Job** | **Job** | **Job** | **Job** | **Job** | **Job** | **Job** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 5 | 1 | 7 | 2 | 8 | 3 | 9 | 4 | 6 | 10 |
| **2** | 7 | 5 | 1 | 6 | 3 | 8 | 4 | 9 | 2 | 10 |
| **3** | 5 | 8 | 6 | 1 | 7 | 2 | 4 | 3 | 9 | 10 |
| **4** | 7 | 5 | 1 | 6 | 3 | 2 | 9 | 8 | 4 | 10 |
| **5** | 5 | 1 | 2 | 8 | 7 | 4 | 6 | 3 | 9 | 10 |
| **6** | 5 | 6 | 7 | 1 | 8 | 9 | 2 | 3 | 10 | 4 |
| **7** | 7 | 1 | 5 | 8 | 2 | 6 | 4 | 3 | 10 | 9 |
| **8** | 5 | 1 | 7 | 8 | 3 | 2 | 6 | 4 | 9 | 10 |
| **9** | 5 | 7 | 6 | 1 | 8 | 3 | 4 | 2 | 10 | 9 |
| **10** | 7 | 5 | 2 | 6 | 1 | 8 | 9 | 3 | 10 | 4 |
| | | | | | | Sum of Weighted Tardiness = **562** | | | | |

Similarly the initial solution for all 18 standard problems such as ABZ5, ABZ6, LA16, etc. has been calculated.

**4.2 Final Solution**

The initial solution obtained from Lekin is feed to the ACO algorithm that was coded in MATLAB, with the following parameters; **$\alpha = 1$, $\beta = 5$, and $\rho_{global} = \rho_{local} = 0.01$**. The final results obtained are shown in table 2.

Table 2: Final results of SB-ACO

| **Problem Type** | **Optimal** | **SBTS** | **GATS** | **SB** | **SB-ACO** |
|---|---|---|---|---|---|
| MT10 | 394 | 466 | 420 | 562 | 404 |
| ABZ5 | 69 | 114 | 82 | 109 | 90 |
| ABZ6 | 0 | 0 | 0 | 0 | 0 |
| LA16 | 166 | 280 | 210 | 184 | 265 |
| LA17 | 260 | 284 | 242 | 268 | 262 |
| LA18 | 34 | 42 | 38 | 83 | 42 |
| LA19 | 21 | 64 | 35 | 55 | 43 |
| LA20 | 0 | 7 | 5 | 6 | 4 |
| ORB1 | 1098 | 1565 | 1190 | 1629 | 1328 |
| ORB2 | 292 | 442 | 320 | 434 | 380 |
| ORB3 | 918 | 978 | 962 | 1104 | 968 |
| ORB4 | 358 | 690 | 360 | 383 | 358 |
| ORB5 | 405 | 591 | 441 | 531 | 463 |
| ORB6 | 426 | - | 447 | 513 | 452 |
| ORB7 | 50 | - | 65 | 142 | 71 |
| ORB8 | 1023 | - | 1108 | 1486 | 1179 |
| ORB9 | 297 | - | 305 | 365 | 305 |
| ORB10 | 346 | - | 380 | 587 | 384 |
| Mean % Error (MPE) | - | 48.36 | 12.44 | 54.0 | 21.74 |

The graphical comparison of different techniques is shown in figure 3.

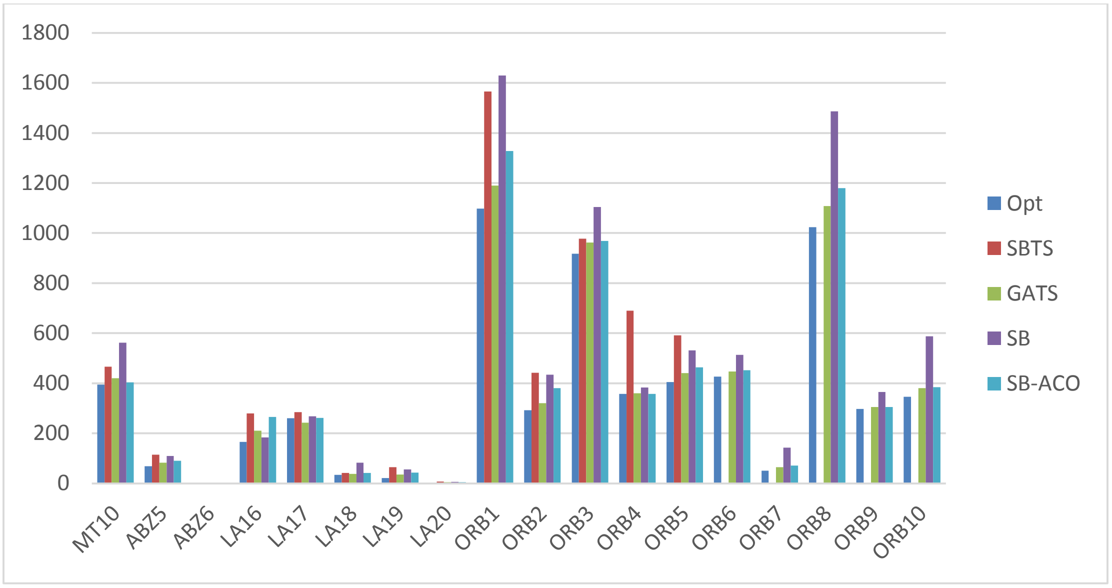


Figure 3: Graphical comparison of different solution techniques

## 5. Conclusions

A successful adaptation of the hybrid metaheuristic SB-ACO for a NP hard job-shop scheduling problem to minimize the weighted tardiness is presented. Our implementation has succeeded in giving robust solutions in short computing times. The results of SB-ACO are better than the SBTS but GATS gives slightly better results. However, GATS has a much larger computing time than the SB-ACO. Therefore, SB-ACO provides a reasonable good solution with much less computing time. Thus, we can conclude that SB-ACO is effective from both the perspective of solution quality and algorithm robustness. The algorithm can serve as an alternative method in the Job Shop Scheduling domain. In the future, we will further fine-tune the ACO parameters to obtain better results.

## Acknowledgements

This research work is supported by NPST Program by King Saud University Project Number 12-INF2574-02.The authors would like to thanks Princess Fatima Alnijris Research Chair for Advanced Manufacturing Technology (FARCAMT), King Saud University for the help and support provided to complete this work.

## References

1. Lenstra J.K., Kan A.H.G.R. & Brucker P., 1977, “Complexity of machine scheduling problems,” Annals of Discrete Mathematics, 1, 343 – 362.
2. Adams J., Balas E., and Zawack D., 1988, “The Shifting Bottleneck Procedure for Job Shop Scheduling,” Management Science, 34(3), 391 – 401.
3. Kolonko M., 1999 “Some new results on simulated annealing applied to the job shop scheduling problem,” European Journal of Operational Research, 113(1), 123-26.
4. Suresh R.K. and Mohanasundaram K.M., 2006, “Pareto archived simulated annealing for job shop scheduling with multiple objectives,” International J. of Advanced Manufacturing Technology, 29(1-2), 184-196.
5. El-Bouri A., Azizi N., and Zolfaghar S., 2007, “A comparative study of a new heuristic based on adaptive memory programming and simulated annealing: The case of job shop scheduling,” European Journal of Operational Research, 177(3), 1894 – 1910.
6. Tanev I.T., Uozumi T., and Morotome Y., 2004, “Hybrid evolutionary algorithm-based real-world flexible job shop scheduling problem: application service provider approach,” Applied Soft Computing, 5(1), 87-100.
7. Gao J., Sun L., and Gen M., 2008, “A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems,” Computers & Operations Research, 35(9), 2892 – 2907.
8. Nowicki E., and Smutnicki C., 2005, “An advanced tabu search algorithm for the job shop problem,” Journal of Scheduling, 8(2), 145 – 149.

9. Haq A.N., Saravanan M., Vivekraj A.R., and Prasad T., 2007, "A scatter search approach for general flowshop scheduling problem," International J. of Advanced Manufacturing Technology, 31(7-8), 731 – 736.
10. Rahimi-Vahed A.R., Javadi B., Rabbani M., and Tavakkoli-Moghaddam R., 2008, "A multi-objective scatter search for a bi-criteria no-wait flow shop scheduling problem," Engineering Optimization, 40(4), 331-346.
11. Sha D.Y., and Hsu C.Y., 2006, "A hybrid particle swarm optimization for job shop scheduling problem," Computers & Industrial Engineering, 51(4), 791–808.
12. Lei. D., 2008, "Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems", International J. of Advanced Manufacturing Technology, 37(1–2), 157–165.
13. Fox B., Xiang W., and Lee H.P., 2007, "Industrial applications of the ant colony optimization algorithm," International J. of Advanced Manufacturing Technology, 31(7–8), 805 – 814.
14. Huang K.L., and Liao C.J., 2008, "Ant colony optimization combined with taboo search for the job shop scheduling problem," Computers & Operations Research, 35(4), 1030 – 1046.
15. Calier, J. and Pison, E., 1989, "An algorithm for solving the job –shop problem," Managing Science, 35, 164-176.
16. Van Laarhoven O., Aarts and Lenstra E.J., 1992, "Job shop scheduling by simulated annealing," Operations Research, 40, 113- 125.
17. Davis L., 1985, "Job Shop Scheduling with Genetic Algorithm," In Proceedings of the First International Conference on Genetic Algorithms, Hilsade NJ, Lawrence Erlbaum Associates, 136-140.
18. Fang H., Ross P. and Corne D., 1993, "A Promising Genetic Algorithm Approach to Job Shop Scheduling," In Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, CA, Morgan Kaufmann Publishers, 375-382.
19. Zhou H. and Feng Y., 2001, "The hybrid heuristic genetic algorithm for job shop scheduling," Computers & Industrial Engineering, 40(3), 191–200.
20. Park B.J., Choi H.R., and Kim H.S., "A hybrid genetic algorithm for the job shop scheduling problems," Computers & Industrial Engineering, l4, 597–613.
21. Wang L. and Zheng D., 2001, "An effective hybrid optimisation strategy for job-shop scheduling problems," Computers & Operations Research, 28(6), 585–96.
22. Goncalves J.F., Mendes J.J.M., and Resende M.G.C., 2005, "A hybrid genetic algorithm for the job shop scheduling problem," European Journal of Operational Research, 167, 77–95.
23. Zhang C., Li P., Rao Y., and Li S., 2005, "A new hybrid GA/SA algorithm for the job shop scheduling problem", Evolutionary Computation in Combinatorial Optimization, 3448, 246–59.
24. Zhang C., Rao Y., and Li P., 2008, "An effective hybrid genetic algorithm for the job shop scheduling problem," International J. of Advanced Manufacturing Technology,39, 965–74.
25. Qing-dao-er-ji R. and Wang Y., 2012, "A new hybrid genetic algorithm for job shop scheduling problem," Computers & Operations Research, 39, 2291-2299.
26. Asano M. and Ohta H., 2002, "A heuristic for job shop scheduling to minimize total weighted tardiness," Computers & Industrial Engineering, 42, 137 – 147.
27. Zhou H., Cheung W., and Leung L.C., 2009, "Minimizing weighted tardiness of job-shop scheduling using a hybrid genetic algorithm," European Journal of Operational Research, 194, 637– 649.
28. Bulbul K., 2011, "A hybrid shifting bottleneck-tabu search heuristic for the job shop total weighted tardiness problem," Computers & Operations Research, 38, 967–983.
29. Singer M. and Pinedo M., 1998, "A computational study of branch and bound techniques for minimizing the total weighted tardiness in job shops," IIE Transactions, 30, 109-118.
30. Bauer A., Bullnheimer B., Richard F. H., and Strauss C., 1999, "An Ant Colony Optimization Approach for the Single Machine Total Tardiness Problem," IEEE, 1445 – 1450.
31. Colorni A., Dorigo M., Maniezzo V., and Trubian M., 1994, "Ant system for Job-shop Scheduling," Belgian Journal of Operations Research, Statistics and Computer Science.
32. Zwaan S., Pais R., and Marques C., 1999, "Ant Colony Optimisation for Job Shop Scheduling," In proceedings of Third workshop on Genetic Algorithms, 1 – 8.
33. Puris A., Bello R., Trujillo Y., Nowe A.Y., and Martínez Y., 2007, "Two-Stage ACO to Solve the Job Shop Scheduling Problem," In Proceedings of CIARP 2007, 447 – 456.
34. Raghavan N.R.S.and Venkataramana M., 2009, "Parallel processor scheduling for minimizing total weighted tardiness using ant colony optimization," Int J Advance Manufacturing Technology, 41, 986 – 996.
35. Besten M., Stiitzle T., and Dorigo M., 2000, "Ant Colony Optimization for the Total Weighted Tardiness Problem," In Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, Publishers Springer-Verlag London, 611 – 620.