International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Using Two Stage Hybrid Algorithm for Solving Flow-Shop Scheduling Problem

A. M. Kadhem

Department of Statistics, College of Economic & Administration, University of Baghdad, Baghdad, Iraq

Abstract: ***The permutation flow shop scheduling is well-known combinatorial optimization problems that have been widely used and many methods have been used to solve this issue because of their widespread use in the business life market. We reject some hybrid methods in solving these issues by generating a range of issues of different sizes. This paper presents a study on using Ant Colony Optimization (ACO), Genetic algorithm (GA) and their combinations (ACO+GA and GA+ACO) to tackle the FSSP. The computation results show that the two-stage algorithms are able to achieve better results in most cases than ACO and GA individually on the FSSP. The proposed two-stage algorithms and visual layout design system provide an effective tool to solve the practical FSSP.***

Keywords: Permutation flow shop scheduling, Ant Colony Optimization, Genetic algorithm, Two stage Algorithm

1. Introduction

Baker (1974) [1] defines the scheduling as the allocation of resources over time to perform a collection of tasks. The resources and tasks may take many forms, for example, "hospital equipment" as resources and "patients" as tasks, usually resources are called (machines) and tasks are called (jobs). A flow shop schedule problem can be stated as follows: there is a set of (M) different machines these machines perform tasks of (n) jobs, each of (n) jobs is processed by (M) machines ($M_1,M_2,...,M_m$) in this order. There are some constraints on jobs and machines, each machine can handle one job at a time and each job can be performed by one machine at a time. The scheduling of manufacturing systems has been the subject of extensive research since the early 1950s'. Many of the applications in the operational research field involve this type of problems. There has been work done on describing different schedule models and environment, on the classification of models and on developing a number of solution procedures. Introductory text books for machine scheduling problems are provided by Conway, Maxwell and Meller (1967) [2] Baker (1974) [1], Franch (1982) [3] and Pindo (1995) [4]. Solving a machine schedule problem means finding the decision that makes us to determine which job should be sequenced first and on which machines. The aim is to find a good (near optimal) or if possible optimal schedule which gives an optimal solution which enables to minimize the time spent on the problem which will minimize the cost for the problem [5].

2. Scheduling Problem (Definition and Classification)

A scheduling problem arises whenever we want to make a daily routine for any planned work [6]. Recently, the m-machine scheduling problem is a widely-studied problem in computer science, see [7] for an excellence survey until 1998, and is concerned with deciding the order for the items to be serviced such as in the manufacturing process described below:

When (n) jobs are processed on (m) machines under given ordering that is given the machine order to process each job, and is given processing times of each job on each machine. The problem is to determine the sequence of (n) jobs on each machine in order to optimize a given objective function, this objective function is generally any non-decreasing function of the completion time of each job, which includes: total elapsed time (make span) to complete the processing of all jobs, weighted mean completion time, weighted mean lateness or tardiness under given due date for each job, and so on. In single stage production system, each job requires one operation, it involves either a single machine or (m) machine operating in parallel, and the jobs are processed by just one machine from several machines that are available [5]. Note that if there is more than one machine and each job j consists of a single operation, this problem will be noted as (parallel machine schedule problem).

3. Flow Shop Scheduling Problem

The flow shop scheduling problem (FSP) is characterized by a set N of n jobs that must be processed by a set M of m machines. All m machines are disposed in series and, without loss of generality, jobs visit machine 1 first, then machine 2 and so on until machine m. Each job needs a given, known in advance, fixed and non-negative processing time at each machine. This is denoted as $p_{i,j}$, for each $j_{2N \text{ and }} i_{2M}$. A job cannot be in process at more than one machine simultaneously and one machine can only process one job at a time

A flow shop schedule problem consists of a set of different machines that perform tasks of jobs. Each of (n) jobs is processed by (m) machines $M_1,M_2,...,M_m$, in this machine order; there are several constraints on jobs and machines:

- There are no precedence constraints among tasks of different jobs.
- Each machine can handle only one job at a time.
- Each job can be performed only on one machine at a time.

The time to process each job "j" on each machine (Mi) is called processing time of job "j" on machine (M_i) and is

expressed by $p_{i,j}$, a sequence of processing times $Pj=(P_{1j},P_{2j},...,P_{mj})$ corresponds to each job "j" and a processing time matrix is defined as an Mmatrix with *Pj* as its column, processing of job "j" on machine M_i is called operation and is expressed by $O_{i,j}$.

4. Permutation Flow Shop Scheduling Problem

The permutation flow shop represents a particular case of the flow shop, having as a goal fined an optimal schedule for (n) jobs on (m) machines. Solving the flow shop problem consists in scheduling (n) jobs (j=1,…,n), on mj machines (i=1,…..,m) . A job consists of (m) operation and the ^{jth} operation of each job must be processed on machine (i), so one job can start on machine (i) if it is completed on machine (i-1) and if machine (i) is free, each operation has a known processing time " pij ". For the permutation flow shop the sequence of the jobs are the same on every machine , if one job is at ^{Jth} position on machine 1, then this job will be at the ^{Jth} position on each machine. Consequently, for the permutation flow shop considering the makespan as objective function to be minimized, solving the problem means determining the sequence of jobs which gives the smallest makespan value (Cmax).

In this case, when there is no passing of jobs allowed, that is, the identity of the job sequence on each machine, the active schedules correspond to the permutation of (n) jobs and so their number is to n! . Each schedule in this case is called a permutation schedule or sequence [6]. Even in the case where passing is allowed in a flow shop, it is sufficient to consider that the job sequence on the first two machines M1, M2 are the same for any normal objective function and moreover the job sequences on the last two machines Mm-1, Mm are the same only when the objective function is the makespan. These facts are stated in the following theorem. Theorem [6]

For the flow shop scheduling problem with any normal objective function to be minimized, it is sufficient to consider that the job sequences on the first two machines are the same and, moreover, the job sequences on the last two machines are the same only when the objective function is the makespan.

As objective function for the sequencing problem, we take the means of the waiting time, competition time, flow time, lateness and tardiness: $\overline{W},\overline{C},\overline{F},\overline{L},\overline{T}$ and their maxima: Wmax, Cmax, Fmax, Lmax, Tmax and also their weighted means by considering the degree of importance of each job are considered. Generally, the minimization of the objective function of the completion time, is called normal objective function [6], in particular, the maximum of the completion times, Cmax=max(C1,C2,…,Cn) is called total elapsed time or makespan.

For calculating the start and completion times of jobs on machines in permutation flowshops, recursive equations are used as follows.

Initialize $q(\sigma i, j)$,the completion time of job i on machine 0,equal to zero. This time indicates the time of availability of a job in the flowshop, and it is equal to 0 for all jobs in case of static flowshops.

For j = 1 to m do

$$q(\sigma i, j) = \max\{q(\sigma, j); q(\sigma i, j-1)\} + tij \quad (1)$$

The flowtime of job i, Ci is given by

$$Ci = q(\sigma i, m) \quad (2)$$

When all jobs are scheduled, the total flow time F ,and the makespan M are obtained as follows:

$$F = \sum_{i=1}^{n} Ci \quad (3)$$

and

$$M = \max\{Ci, i = 1,2,..,n\}.$$

It is to be noted that $q(\phi, j)$ is equal to 0 for all j, where ϕ denotes a null schedule.

5. Ant Colony optimization

(ACO) was suggested as a new heuristic method to solve optimization problems by Dorigo and Gambardella [9]. The reformed form of the (AS) algorithm and functions is shown as follows. Each ant generates a complete solution by choosing the nodes according to a probabilistic state transition rule. The state transition rule is given in (5.1) is called a pseudorandom-proportional rule:

$$P^k(i,j) = \frac{(t_{i,j})^{\alpha} \quad (\eta_{i,j})^{\beta}}{\sum_{j \in N_i^k} (t_{i,j})^{\alpha} \quad (\eta_{i,j})^{\beta}}$$

Where tij is the amount of pheromone in edge ij, ηij = 1/δij where δij is the cost of edge ij, α and β are parameters that determine the relative importance of η versus t ,and Nik is the remaining node set of ant k based on moving from node i to build a feasible solution [10] .

The parameters α, β are user defined parameters that determine the degree to which the pheromone is used versus the heuristic distance in deciding where to move. Setting β = 0 will result in only the pheromone information being used whereas if α = 0, only the heuristic information will be used [11].

In either case in ACO, only the globally best ant that has built the best solution deposits pheromone in the graph. At the end of an iteration of the algorithm, once all the ants have built a solution, pheromone is added to the arcs used by the ant that found the best tour from the beginning of the trial. This updating rule is called the global updating rule of pheromone

$$tij = (1-p)tij + p.\Delta tij$$

where 0 <p<1 is a pheromone decay parameter and Δtij equals to

$$\Delta t_{ij} = \begin{cases} \frac{1}{best\ cost} & if\ (i,j) \in best\ sequence \\ 0 & otherwise \end{cases}$$

In ACO, ants perform step-by-step pheromone updates using local updating rule of pheromone. These updates are performed to favor the emergence of other solutions than the best so far. The updates result in step-by-step reduction of the pheromone level of the visiting edges by each ant.

The local updating rule of pheromone is performed by applying the rule:

$$\tau_{ij} = (1 - \zeta).t_{ij} + \zeta.t_0$$

τ_0 is a small fixed value and $0 < \zeta < 1$ is the local evaporation of pheromone [10]

The ACO structure is shown in the following Algorithm procedure:

ACO algorithm procedure

```
1   Set pheromone trails to be small constant
2   While (termination condition not met)
3       Place each ant on initial node (its index usually)
4       Repeat
5           For each ant do
6               Chose next node by Apply State Transition Rule
7       End For
8       Until "each ant build one a solution"
9       Chose the best solution
10      Apply Local Update pheromone
11      Apply Global Update
12  End While
```

6. Genetic Algorithm (GA)

Genetic Algorithms (GA) were originally proposed by John H. Holland [12]. They are search algorithms that explore a solution space and mimic the biological evolution process.

Genetic algorithms work with the population of solution each solution is represented as a string the (GA) technique based on the mechanism of evolution .The solution space is usually represented by a population. New structures are generated by applying simple genetic operators such as (select, cross-over, and mutation). The members with higher fitness values (i.e., better objective function values) in the current population will have higher probability of being selected as parents, which is similar to Darwin's concept of survival of the fittest. The initial population is randomly generated, which means that the optimality of the final solution would not be guaranteed. Therefore, in the initial population, at least one solution having the minimum makespan (objective function of our problem) is included applying (select, cross-over and mutation), to generate new population and save the best solution in every generation. The best one from saved solutions becomes GA solution [13], the fitness value of a solution is a vector representing the function values (makespan). A parent is generated by selecting the best solutions from the current population. Then, solutions with good fitness values in each population are selected and recombined in each generation to produce a new offspring after applying the genetic operators for each new offspring we get a new population. We note that the mutation operation (for example) is based on the pairwise interchange (swap) of two jobs in the corresponding sequence. There are several applications of Genetic Algorithms (GA) have been widely applied to various fields since 1975. They are applied to business, scientific, and engineering areas including:

7. Basic Stricture of Genetic Algorithm

The main components of a genetic algorithm are as follows [14] :

1) **Solution Encoding**
   A chromosomal representation of solutions, (solution encoding).
   For the machine schedule problem, the natural permutation representation of a solution is a permutation of the integers 1,…,n, which defines the processing order of n jobs. Each chromosome is represented by such a scheduling solution, i.e., the natural permutation representation of a solution.
2) **Initial Population**
   The creation of an initial population of chromosomes, (initial population).
   In order to approximate an optimal solution as near as possible, the initial population of chromosomes is created by scheduling heuristic dispatching rules (heuristics methods), combined with random methods.
3) **Fitness** (evaluation)
   The measurement of chromosome fitness is based on the objective function (fitness).When a population is generated, each chromosome is evaluated and its fitness is calculated for each chromosome. Finally each chromosome is assigned its fitness value along of the population size.
4) **Selection**
   Natural selection of some chromosomes, by selection methods (according fitness value usually), chromosomes (parents) are selected from the population for combining to produce new chromosomes (children), i.e., for applying genetic operators.
5) **Genetic Operators**
   Genetic Operators (crossover and mutation) applied to the chromosomes whose role is to create new members, i.e., children, in the population by crossing the genes of two chromosomes (crossover operators) or by modifying the genes of one chromosome (mutation operators):
   a) **Crossover:**
      The role of a crossover operator is to combine elements from two parent chromosomes to generate one or more child chromosomes.
   b) **Mutation:**
      The role of a mutation operator is to provide and maintain diversity in a population so that other operators can continue to work.

6) **Replacement**
Natural selection of the members of the population, who will survive (replacement), is based on elitism. That is to keep the best chromosomes of the current population and their offspring. They will form a new population to survive into the next generation.

8. Parameter Selection

Natural convergence of the whole population that is globally improved at each step of the algorithm. For choosing suitable values of parameters such as population, size crossover and mutation.

The performance of a (GA) depends largely on the design of the above components and the choice of parameters such as population size, probabilities of genetic operators (i.e., crossover and mutation), and number of generations.

The following steps give us the outline of (GA):

Genetic algorithm procedure

1- **initialization** : create the initial population
evaluate this population
save the best element from this population as a {(GA) solution}
2- **While** stopping condition is not satisfied do
select a good solutions (parents) form current population
generate new population by genetic operators (crossover and mutation)
evaluate this population
if a new best individual element is found save it as a {(GA) solution}
3- **end while**

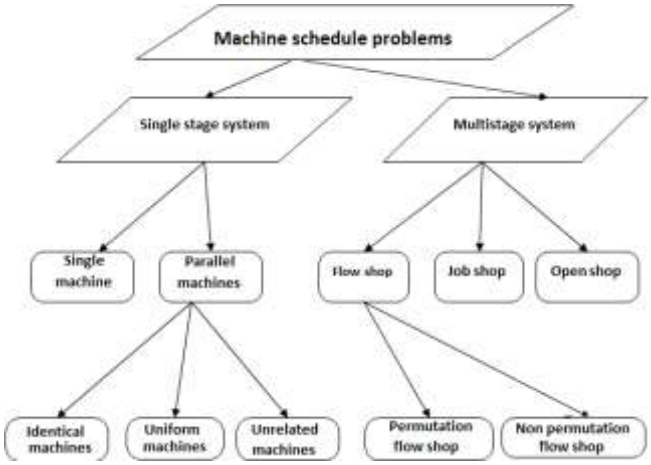

Figure 1: Taxonomy for machine scheduling problem

9. The Two-Stage Approaches

The ACO and GA present two different ways to solve the flow shop scheduling FSS, and a combination of two or even more approaches may give us a better chance to find the optimal solution. Therefore we propose two two-stage approaches in this place: ACO +GA, and GA+ACO.

In the first two-stage approach ACO +GA, we apply TS to improve the randomly generated particles and then use ACO to make further improvement. In the second two-stage approach

ACO+GA, we first use ACO to find the best solution and then use GA to make further improvement.

10. Experimental Result

The presented GA, ACO, and two-stage approaches are implemented using Matlab 2015. The program was run on a computer with a CORE I5 CPU 2.6 GHz, 4GB memory. We generate some problem with different number of product and different number of machine.

Table 1: Comparison between the algorithms based on best criteria

m	n	ACO	GA	ACO+GA	GA+ACO
3	5	243.00	249.00	252.00	243.00
	10	422.00	424.00	446.00	421.00
	15	623.00	638.00	638.00	623.00
	20	848.00	855.00	869.00	848.00
	25	1026.00	1026.00	1058.00	1026.00
	30	1139.00	1157.00	1176.00	1139.00
	35	1331.00	1338.00	1357.00	1325.00
	40	1509.00	1525.00	1542.00	1504.00
	45	1696.00	1725.00	1727.00	1690.00
	50	1833.00	1854.00	1854.00	1827.00
4	5	266.00	266.00	284.00	266.00
	10	471.00	476.00	478.00	470.00
	15	607.00	623.00	631.00	608.00
	20	779.00	793.00	796.00	779.00
	25	998.00	1027.00	1012.00	999.00
	30	1187.00	1199.00	1202.00	1179.00
	35	1403.00	1428.00	1434.00	1398.00
	40	1565.00	1571.00	1600.00	1560.00
	45	1703.00	1720.00	1748.00	1695.00
	50	1871.00	1894.00	1883.00	1867.00
5	5	284.00	284.00	300.00	284.00
	10	511.00	503.00	522.00	509.00
	15	644.00	646.00	659.00	639.00
	20	895.00	901.00	947.00	890.00
	25	1087.00	1121.00	1119.00	1085.00
	30	1223.00	1252.00	1260.00	1222.00
	35	1389.00	1415.00	1416.00	1387.00
	40	1571.00	1607.00	1613.00	1566.00
	45	1802.00	1848.00	1846.00	1798.00
	50	1920.00	1926.00	1944.00	1916.00

Table 1,2,3 above shown the comparative between the algorithms and instances depending on the performance best, worst, Std. and time the results shown that the superiority of the algorithms (GA-ACO and ACO-GA) on the machine 3 for all jobs except jobs (45 and 50). In the machine 4 we can show that the algorithm ACO has the best value depending on the time, on jobs (45 and 50) algorithm (GA) show the best and for all the jobs in the machine (4) the hybrid algorithm (GA-ACO and ACO-GA) have the best value depending on the time also. At least, in the machine (5) depending on the time the algorithms ACO and hybrid have the same time in the job (5). Also in the same machine we can see that the algorithm GA has the best time compare with the (GA-ACO and ACO-GA) and ACO.

Table 2: comparison between the algorithms based on worst criteria

m	n	ACO	GA	ACO+GA	GA+ACO
3	5	243.00	252.00	282.00	243.00
	10	425.00	447.00	457.00	424.00
	15	625.00	648.00	667.00	623.00
	20	848.00	889.00	889.00	853.00
	25	1030.00	1041.00	1078.00	1027.00
	30	1146.00	1182.00	1245.00	1146.00
	35	1337.00	1382.00	1375.00	1328.00
	40	1519.00	1571.00	1584.00	1517.00
	45	1701.00	1752.00	1779.00	1697.00
	50	1835.00	1914.00	1894.00	1839.00
4	5	266.00	284.00	303.00	266.00
	10	474.00	502.00	507.00	474.00
	15	616.00	642.00	643.00	616.00
	20	789.00	818.00	824.00	791.00
	25	1010.00	1051.00	1060.00	1010.00
	30	1191.00	1229.00	1270.00	1191.00
	35	1417.00	1481.00	1464.00	1410.00
	40	1577.00	1612.00	1648.00	1566.00
	45	1709.00	1766.00	1800.00	1714.00
	50	1882.00	1940.00	1967.00	1878.00
	5	284.00	300.00	317.00	284.00
5	10	516.00	534.00	573.00	511.00
	15	647.00	687.00	705.00	644.00
	20	910.00	949.00	989.00	908.00
	25	1100.00	1146.00	1148.00	1097.00
	30	1238.00	1276.00	1294.00	1232.00
	35	1398.00	1447.00	1451.00	1393.00
	40	1584.00	1634.00	1660.00	1585.00
	45	1817.00	1928.00	1891.00	1814.00
	50	1930.00	2008.00	1975.00	1925.00

Table 3: Comparison between the algorithms based on time and standard deviation

m	n	ACO		GA		ACO+GA		GA+ACO	
		STD	T(s)	STD	T(s)	STD	T(s)	STD	T(s)
3	5	0.00	0.34	1.41	0.81	10.78	0.22	0.00	0.18
	10	1.14	0.36	9.40	0.70	4.45	0.40	1.30	0.29
	15	0.89	0.61	5.05	0.84	12.14	0.66	0.00	0.45
	20	0.00	0.88	14.87	0.97	10.95	0.92	2.24	0.63
	25	2.05	1.21	5.57	1.11	7.97	1.24	0.45	0.83
	30	3.03	1.52	10.05	1.24	27.04	1.59	2.77	1.04
	35	2.83	1.93	17.46	1.38	6.91	2.01	1.30	1.28
	40	4.10	2.35	16.99	1.52	16.62	2.42	5.22	1.53
	45	1.95	2.76	11.33	1.67	22.29	2.85	2.74	1.79
	50	0.71	3.17	22.26	1.80	15.32	3.25	5.20	2.05
4	5	0.00	0.53	7.36	2.08	8.35	0.64	0.00	0.56
	10	1.52	1.23	10.68	2.44	10.45	1.48	1.58	1.08
	15	3.94	2.28	6.91	3.14	5.02	2.34	3.46	1.69
	20	4.16	2.95	9.89	3.46	11.12	3.52	4.93	2.29
	25	4.85	4.60	9.15	4.10	18.19	4.65	3.96	3.14
	30	1.79	5.92	11.24	4.71	28.78	6.12	5.59	3.97
	35	4.97	7.27	21.52	5.16	14.64	7.53	5.22	4.67
	40	4.85	8.62	16.98	5.79	21.68	9.07	3.00	5.74
	45	2.28	10.07	17.74	6.30	19.32	10.59	7.44	6.53
	50	4.92	11.88	19.23	6.77	30.83	12.06	4.51	7.60
5	5	0.00	0.59	6.83	2.03	7.65	0.64	0.00	0.59
	10	2.17	1.03	12.12	2.08	20.19	1.10	0.89	0.91
	15	1.41	2.30	16.36	3.02	17.94	2.16	2.24	1.54
	20	5.94	3.59	18.51	3.75	17.43	3.51	7.01	2.47
	25	5.37	4.60	9.37	4.23	11.94	4.70	5.17	3.17
	30	5.85	5.73	9.42	5.04	14.60	6.20	3.63	4.01
	35	4.44	7.07	12.18	5.34	13.96	7.42	2.51	4.69
	40	5.07	8.93	10.69	6.06	17.47	9.23	6.91	5.78
	45	6.02	7.27	31.94	4.80	18.22	6.85	6.63	4.45
	50	4.42	3.45	32.87	2.03	12.24	3.55	3.74	2.22

11. Conclusions

In this paper, we proposed to use GA, ACO and their combinations to solve the flow shop scheduling problem (FLP) with size constraints on a continual planar site. The computational results of the experiments suggest that GA+ACO give us the better result than GA and ACO and ACO+GA. It is possible to demonstrate the possibility of obtaining good results for solving complex problems, thereby integrating the advantages of two techniques and thus raising the exploration within the search space, and also raising the convergence of it to obtain universal optimization at a reasonable time

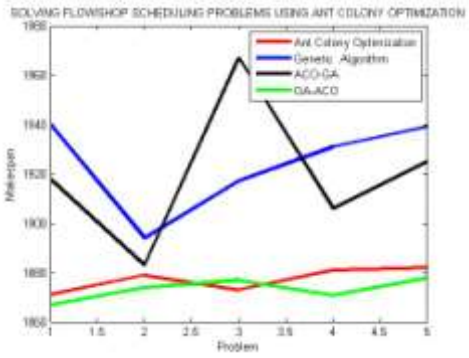

Figure 2: Comparison between algorithms when m=3

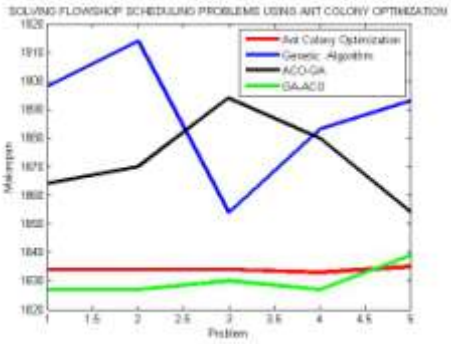

Figure 3: Comparison between algorithms when m=4

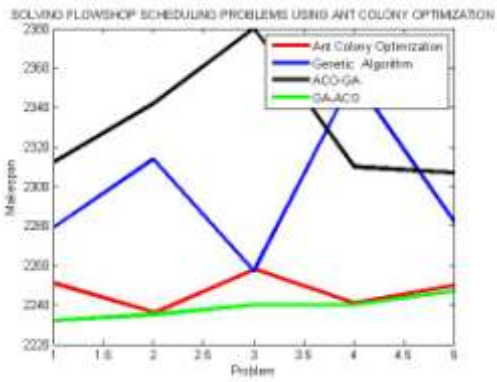

Figure 4: Comparison between algorithms when m=5

References

[1] Baker, K.R "Introduction to Sequencing and Scheduling " Wiley New York, (1974)
[2] Belman R., Eesogbue A.O., Nabeshima I., "Mathematical Aspects of Scheduling and Application", pergamon press, Oxford, New York, Toronto, Sydney, Paris, (1982)
[3] Al-Samarii N. A.A., "Signatures verification using neural network", M.Sc. thesis, College of Engineering, University of Al-Mustansiriyah, April -2004
[4] Chen B., Potts C., and Woeginger G.J., "A Review of Machine Scheduling: Complexity Algorithms and Approximability", handbook of combinatorial optimization. (1998).
[5] Chen C.L., Vempati V.S., and Aljaber N. , " An application of genetic algorithms for flow shop problems" European Journal of operation research , 80 389-396, (1995).
[6] Conway R.W., Maxwell W.L., and Miller L.W. "Theory of Scheduling" Addison Wesley, Reading, MA. (1967)
[7] Cook S.A. "The Complexity of The Theorem-Proving procedure", proc. 3rd annual ACM Symp. Theory compute, 151-158 (1971).
[8] Crauwels, H. "A comparative study of local search methods for one machine sequence problem". Ph.D. thesis Katholieke University, Heverlee. Belgium (1998).
[9] Dorigo M., and Gambardlla, L.M., "Ant algorithms for discrete optimization", Massachusetts Institute of technology, artificial life 5: 137-172 (1999).
[10] Franch, S. "Sequencing and Scheduling an Introduction to Mathematics of Job Shop" , John Wiley & Sons, New York (1982).
[11] Gupta J.N.D., Hennig K., Werner F., "Local search heuristics for two-stage flow shop problems with secondary criterion ", Ball stat university, Muncie, IN43,306, USA. Pergamon computer and operation research 29, 123-149,(2002).
[12] Holland J. H. "Adaptation in Natural and Artificial Systems". Ann Arbor, University of Michigan Press, 1975.
[13] Johnson S.M., "Optimal Two and Three Stage Production Schedule with Setup Time Included", Nav.Res.Log.Quart.1No1 (1954)
[14] Keivan G. and Fahimeh M., "ACS - TS: train scheduling using Ant Colony system", Journal of applied mathematics and design sciences. Article ID (95060) 1-28, (2006).
[15] Lee J.K. and Kim Y.D., "Search heuristic for resource constrained project scheduling", Journal of the operation research society 47, 678-689 (1996).
[16] Lenstra J.K., Rinnooy Kan A.H.G., Bruck B., “Complexity of machine scheduling problems", Annals of discrete math. 1, 343-362, (1977).
[17] Liu N., Mohamed A. Abdelrahman, and Srini Ramaswamy," A Genetic Algorithm for the Single Machine Total Weighted Tardiness Problem", Tennessee Technological University, Cookeville, TN 38505, USA, 2003
[18] Pindo, M. "Scheduling Theory Algorithms and System" prentice hell, Inc., Englewood cliffs, New Jersey, (1995).
[19] Reeves C.R.," A Genetic algorithm for flow shop sequencing ", computer and operation research, 22, 5-13 (1995).
[20] Ventresca M., and Ombuki B.M.. "Ant Colony Optimization for Job Shop Scheduling Problem" , Brock university Canada, L25, 3A1, (2004).