SOME NOVEL METHODS FOR FLOW SHOP SCHEDULING

ROBIN KUMAR SAMUEL*, P.VENKUMAR[2]
[2] Department of Mechanical Engineering, Kalasalingam University, Krishnancoil, Tamil Nadu, India
* Department of Mechanical Engineering, Noorul Islam University, Kumaracoil, Tamil Nadu, India
Email : robn2121@yahoo.co.in
Ph : +91-9487689453

ABSTRACT

The permutation flow shop scheduling problem (PFSP) is one of the most well known and well studied production scheduling problems with strong industrial background. Most scheduling problems are combinatorial optimization problems which are too difficult to be solved optimally, and hence heuristics are used to obtain good solutions in a reasonable time. The specific goal of this paper is to investigate scheduling heuristics and find the minimum makespan. The paper demonstrates the various techniques developed to find the minimum makespan starting from the simple Direct Method(DA), Genetic Algorithm(GA), Simulated Annealing (SA) and Hybrid metaheuristic using SA with GA. The effect of combining fast polynomial reinsertion algorithm such as NEH to SA is also presented in this paper. The performance of the heuristics is compared relative to each other on a set of benchmark test problems of Carlier, Reeves and Tailbard.
Keyword : *Flow shop, Makespan , Direct method, Genetic Algorithm, Simulated Annealing, Hybrid Metaheuristic*

1.INTRODUCTION

Scheduling is the allocation of resources (e.g. machines) to tasks (e.g. jobs) in order to ensure the completion of these tasks in a reasonable amount of time. The flow shop scheduling problem (FSP) is defined as given a set of n jobs, $J_1.J_2,\ldots..J_N$, to be processed by m machines $M_1,M_2,\ldots.M_M$. Each job demands m operations and every job has to obey the same operational flow i.e. job J_K for K = 1,2,…N is processed first in machine M1, then in machine M2 and so forth up to M_M. If a job J_K does not use all the machines its processing flow continuous to be the same but with the time zero whenever that happens. A machine processes just a single job and once it is started it cannot be interrupted up to its completion. It is worth of mentioning that the total search space of possible sequences to be considered is very large. The problem of finding optimum processing sequence is known as NP-hard. The flow shop problem is subjected to a additional constraint that no in-process waiting is permitted.

In FSP input is given by a matrix P with m x n non – negative elements, where P_{ik} is associated with job J_k processing time on machine M_i. A lot of heuristics have been proposed in order to provide a good approximate solution in a reasonable time. Those include branch and bound techniques, various nature based metaheuristics such as, genetic algorithms, tabu search and simulated annealing.

2. LITERATURE REVIEW

The makespan criterion for flow shop scheduling has been thoroughly studied since Johnson's seminal paper back in 1954. Nevertheless, it is still a hot topic of research .Exact approaches have shown limited results for medium or large problems and are specially sensible to the number of machines .There is a wealth of literature on heuristic and metaheuristic methods for the PFSP problem and makespan criterion. A survey of literature shows that most of the heuristics for flow shop scheduling aim at minimizing makespan (MS). Johnson's algorithm [1] is the earliest known optimal solution algorithm for n jobs two machines flow shop scheduling problems. Palmer [2] proposed a slope order index to sequence the jobs on the machines based on the processing time. The heuristic proposed by Campbell, Dudek and Smith (CDS) [3] is basically an extension of Johnson's algorithm. It builds $m - 1$ schedules by clustering the m original machines into two virtual machines and solving the generated two machine problem by repeatedly using Johnson's rule.

Nawaz [4] heuristic algorithm is based on the assumption that a job with high total processing time on all the machines should be given higher priority than job with low total processing time. The NEH algorithm does not transform the original m machine problem in to one artificial two-machine problem. It builds the final sequence in a constructive way, adding a new job at each step and finding the best partial solution. Rajendran [5] in his improvement heuristic takes the first seed from CDS algorithm. A heuristic preference relation is proposed and is used as the basis to restrict the search for possible improvement in the multiple objectives. Temiz and Erol [6] propose a fuzzy branch and bound algorithm approach for the flow shop scheduling problem. The branch and bound algorithm of Ignall and Schrage is modified and rewritten for three-machine flow shop problems with fuzzy processing times. Fuzzy numbers is used to determine the minimum completion time. Proposed algorithm gets a scheduling result with the membership function for the final completion time. Koulamas [7] reported a new two phase heuristic, Heuristic Flow shop scheduling with C_{max} objective (HFC). In the first phase, the HFC heuristic makes extensive use of Johnson's algorithm. The second phase improves the resulting schedule from the first phase by allowing job passing between machines, i.e. by allowing non-permutation schedules. Suliman [8] developed a two phase improvement heuristic. In the first phase, a schedule is generated with the CDS heuristic method. In the second phase, the schedule generated is improved with a job pair exchange mechanism with a directionality constraint. The solutions obtained are compared with the solutions obtained by the NEH sequencing method. The comparison shows similar performance by the two methods. Dipak Leha and Mukerjee [9] proposed a genetic algorithm based methodology with a view toward minimising the makespan of the schedule generated in a flow shop environment. Minimization of makespan results in maximization of overall resource utilisation. This algorithm is tested computationally and compared with the best existing heuristic algorithm. It yields a superior solution but the execution time is higher. Chen et al. [10] generated a genetic algorithm based heuristic for flow shop problems and compared the computational results of the heuristic with the results of existing heuristics. In the present work the following algorithms have been developed i.e. Direct Method Algorithm (DMA), Genetic Algorithm (GA), Simulated Annealing and Hybrid SA-GA. NEH algorithm is integrated with SA, The solution of NEH is given as a starting sequence to simulated annealing to obtain the final solution. The performance of the proposed algorithm is demonstrated by applying it to benchmark problems available in the OR-Library.

3. Problem Description

The flow shop problem can be formulated as follows. Each of n jobs from the job set $i = \{1, 2, \ldots, n\}$, for $n > 1$, has to be processed on m machines 1,2,….,*m* in the order given by the indexing of the machines. Thus job *j*, *j* € *J* , consists of sequence of m operations; each of them corresponding to the processing of job j on machine i during an uninterrupted processing time $p_{i\ j} \geq 0$. It is assumed that a zero processing time on a machine corresponds to a job performed by that machine in an infinitesimal time. Machine *j*, *j* = 1, 2, ...,*m*, can execute at most one job at a time, and it is assumed that each machine processes the job in the same order . The objective is to find a sequence for the processing of the jobs on the machines so that the total completion time or makespan of the schedule (C_{max}) is minimized. The processing times needed for the jobs on the machines are denoted as *pi j* , where $i = 1,\ldots,n$ and $j = 1, \ldots, m$; these times are fixed, known in advance and non-negative. There are several assumptions that are made regarding this problem:

- Each job *i* can be processed at most on one machine *j* at the same time.
- Each machine *m* can process only one job *i* at a time.
- No preemption is allowed, i.e. the processing of a job *I* on a machine *j* cannot be interrupted.
- The set-up times of the jobs on machines are included in the processing times.
- The machines are continuously available.
- In-process inventory is allowed. If the next machine on the sequence needed by a job is not available, the job can wait and join the queue at that machine.

4. Algorithms Developed

4.1 Direct Method Algorithm (DMA)

The direct method algorithm as the name specifies is the most direct approach to calculate the makespan. It is an exact algorithm and gives accurate results. In spite of the advantage of giving accurate results the DMA cannot be used for solving problems of size grater then 20 jobs. The drawback of this algorithm lies in the fact it has to ennumarate all possible sequences to find the best sequence thus the time consumption exponentially increases with the size of the problem. The working of DMA is given demonstrated.

Step 1 – Read the given input (N X M Matrix)

Step 2 – Generate a set of sequence of N! set.

Step 3 - Calculate the makespan for each generated sequence.
Step 4– Compare all the makespan and find the sequence with minimum makespan .
Step 5 – Display sequence with minimum makespan.

4.1.1. Computational Results for DMA

The developed algorithm is evaluated on a large number of problems. Various size problems have been generated randomly and also the algorithm is tested on benchmark problems contributed to the OR-library in which
9 problem named rec01, rec03,…..rec17 respectively are given by Reeves and other 8 problems named car1, car2, ….., car8 are obtained from Carlier. The algorithm is programmed in Turbo C++ and runs on a PC with Intel Pentium 2.2GHz and 3GB RAM. The developed algorithm gives accurate result and outperforms genetic algorithm in some cases An example for such an instance is the 5 job 10 machine problem earlier solved by metaheuristics which gave the makespan as 862 [11], the developed DMA in this paper gives the makespan as 859 for the same problem. The sequence given by the metaheuristic was J4 J3 J2 J5 J1. The sequence obtained from direct method algorithm J2 J4 J3 J5 J1 and J4 J2 J3 J5 J1. Although the developed algorithm is accurate but the computational time required to solve the problem is very high. From the results it can be noted that the DMA can be used only for problems of range up to 17 jobs problem for which the computation time is 100 sec for larger problems the computational time drastically increases such as 1hr 30 min in case of 20 jobs and 5 machine problem and 4 hrs 40 min in case of 20 jobs 10 machine problem and so on.

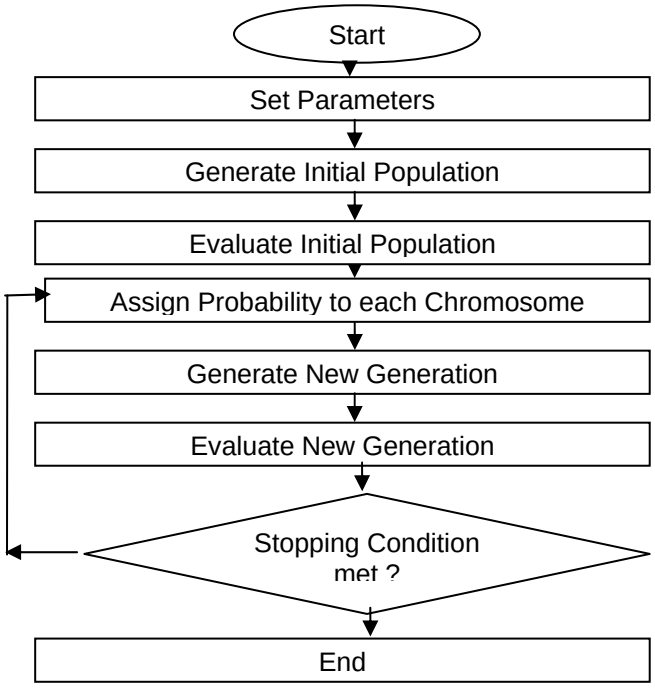
4.2 GENETIC ALGORITHM

Genetic algorithms are an approach to optimization based on principles of biological evolution. Genetic algorithms maintain a population of possible solutions to a problem, encoded as chromosomes based on a particular representation scheme. After generating an initial population, new individuals for this population are generated via the process of reproduction. Parents are randomly selected from the current population for reproduction with the better ones (according to the evaluation criteria) more likely to be selected. The genetic operators of mutation and crossover generate children (i.e., new individuals) by random changes to a single parent or combining the information from two parents respectively. Genetic algorithms have been applied to scheduling problems in a wide variety of domains including Jobshop/Flowshop scheduling , Urban transit systems, Supply chain management, Exam timetabling , Scheduling computing tasks , Scheduling laboratory equipment.

The reason for genetic algorithms success at a wide and ever growing range of scheduling problems is a combination of power and exibility. The power derives from the empirically proven ability of evolutionary algorithms to efficiently and globally competitive optima in large and complex search spaces. The favorable scaling of evolutionary algorithms as a function of the dimension of the search space makes them particularly effective in comparison with other search algorithms for the large search spaces typical of real world scheduling. The exibility of genetic algorithms has multiple facets. Even the standard genetic algorithm (i.e.,bit string representation with traditional crossover and mutation operators) can effectively handle problems that many traditional optimization algorithms cannot including: (i) discrete spaces, (ii) nonlinear, discontinuous evaluation functions, and (iii) nonlinear, discontinuous constraints. The use of nonstandard genetic algorithms and the tailoring of representation, operators, initialization method, etc. to the problem/domain greatly increases the range of problems to which genetic algorithms can be effectively applied. The GA developed in this paper uses Rank order selection procedure to assign the fitness value to the chromosomes (individual solution), the route wheel selection procedure which is generally used is avoided because many of the Chromosomes may have very high makespan value thus it would result in selection of chromosome of high makespan. Along with rank order selection partially matched crossover operator and shift mutation operator is used. As GA may lead to premature convergence, to come out of it the shift mutation is used. If for a specified number of generations, there is no improvement in the solution then, the GA again restarts with the available best solution.
Rank Order Formula

$$\text{Expected value}(i,t)=(\min)+(\max-\min)X\ \frac{rank(i,t)-1}{N-1}$$

Where N = Population size = 50, Min = 0.4 ; Max = 1.6
The algorithm uses Permutation Encoding In this, every chromosome is a string of numbers, which represents number in a sequence.



Crossover

The single point crossover is used One crossover point is selected, till this point the permutation is copied from the first parent, then the second parent is scanned and if the number is not yet in the offspring it is added. The crossover probability is fixed as 0.6, the probability is fixed based on trail and error bases and the algorithm was tuned to 0.6.

* cross over point

Table 1.1. – GA Crossover

P1	1	2	3	4	5	6	7	8	9
P2	4	5	3	6	8	9	7	2	1
C1	1	2	3	4	5	6	8	9	7
C2	4	5	3	6	8	7	9	1	2

Mutation

Genetic algorithm is prone to get struck in a particular local region. Thus to avoid this the mutation operation is performed. Mutation changes randomly the new offspring by interchanging some digits. The mutation probability is fixed as 0.015; thus if the probability is less then equal to 0.015 then the mutation operation is performed. In this paper, mutation operators called SWAP is used, two distinct elements are randomly selected and swapped.

Before Mutation								
1	2	3	4	5	6	7	8	9
After Mutation								
1	8	3	4	5	6	7	2	9

Table 1.2. – GA Mutation process

The process is repeated again and again to get new better generations. After fine tuning the values of parameters were set as follows.
Population size = 50.
Probability of crossover = 0.6
Probability of mutation = 0.015
Maximum No. of generations = n x m
No of generation to restart the GA if solution is not improved = 50

Stopping Condition

Theoretically, to guarantee the convergence of a GA, a long enough Markov chain is required , which may lead to a large computation requirement. As there is no practicable rule to set a suitable stopping condition, the usual way is to set a maximum number of generations. Since the solution length depends on the number of jobs, and the machines may indirectly affect the search quality and efficiency, we set the maximum generation to *N x m*, which may provide a good compromise between solution quality and search efficiency

4.2.1. Computational Result for GA

Computational simulation is often carried out with some benchmarks. In this paper, we select 29 problems contributed to the OR-Library by D. C. Mattfeld and R. J. M. Vaessens , where 21 problems named rec01, rec03, . . ., rec41, respectively, are given by Reeves [8] , the other 8 problems named car1, car2, . . ., car8 can be obtained from Carlier . The result for the problems are consolidated in table 1 It is found that the genetic algorithm solves all the given problem in much lesser time then the DMA. The results obtained using the GA are not as accurate as the results obtained using DMA but GA is able to solve the bigger problems which DMA fails to do. GA is an approximate method whearas DMA gives the exact solution but only for problems of smaller size.

4.3. SIMULATED ANNEALING

Simulated annealing (SA) is motivated by an analogy to annealing in solids. The idea of SA comes from a paper published by Metropolis et al. in 1953 [12]. Metropolis algorithm simulated the material as a system of particles. The algorithm simulates the cooling process by gradually lowering the temperature of the system until it converges to a steady, frozen state. Kirkpatrick et al. [13] took the idea of the Metropolis algorithm and applied it to combinatorial and other optimization problems. The major advantage of SA over other methods is an ability to avoid becoming trapped at local minimum. The algorithm employs a random search, which not only accepts changes that improve objective function but also some changes that do not improve it. SA is a variation of hill climbing in which, during search process, some nonimproving moves may be made. SA first generates neighborhood solutions from a current solution and move to one of them until stopping criteria is met. The performance of SA is influenced by some factors such as initial temperature (T_0), final temperature (T_f), number of temperature decline (N), and number of iterations at each temperature (Nlimit). SA begins with an initial solution (IS) and, after evaluating it by means of a cost function, a new neighborhood is generated. This new neighborhood solution is accepted when it improves the value of cost function. In a condition where the value of cost function of new neighborhood solution is worse, it also can be accepted if it satisfies the following criteria:

$$r > \exp - (\Delta E \backslash K T)$$

where r is generated using uniform distribution between 0 and 1; ΔE is the difference between the value of the cost function of the current solution and the new neighborhood solution, and T is the current temperature. The initial temperature is set as 1200. At each temperature 1000 random sequences are generated. The temperature is decresed by an factor of 0.9. The value of K is determined by trail and error method. The performance of the algorithm for various K values are consolidated in the table 2. The value of K is taken as 0.04 for which the algorithm shows the best results.

Parameters for SA

Initial Temperature – 1200
Temperature reduction factor – 0.9
Number of sequences at each temperature – 1000
K – 0.04

In order to increase the performance of SA an effective fast polynomial NEH algorithm is used along with SA. The NEH algorithm generates a sequence based on the fact that the jobs with maximum processing time should be given preference for the first place in the sequence. The sequence obtained from NEH is given as input to SA and it becomes the first sequence of SA, then the algorithm proceeds in the normal manner.

4.3.1. Computational Result for NEH+SA

The NEH+SA algorithm was tested on the same set of problems given by Carlier and Reeves as used for testing GA. The algorithm was run on a intel core 2 duo processor, the algorithm was coded in c++. The results obtained are summarized the table 1 and the comparison is also shown in the chart. It was found that NEH+SA outperformed GA in all the cases and even in some cases it was able to provide a better solution then the most optimum found so far, but in most of the cases the algorithm was able to achieve a value closer to the optimum one.

4.4. HYBRID METAHEURISTIC

In contrast to the early days of metaheuristic research, the last 5-10 years have produced a large number of algorithms that simply do not fit into a single metaheuristic category. This is because these untraditional approaches combine various algorithmic ideas, often originating from several branches of artificial intelligence, operations research and computer science in general. Such approaches are commonly referred to as hybrid metaheuristics . In this research we hybridize the simulated annealing with genetic algorithm. The hybrid algorithm consist of two phase, the first phase consist of Simulated annealing combined with NEH algorithm. The result obtained in the first phase is given as input into the second phase which is Genetic Algorithm. An initial random population is generated along with this population the sequence obtained as output in first phase is also included in the initial population. The gantt chart for various problems are presented in Fig 2.1and 2.2.

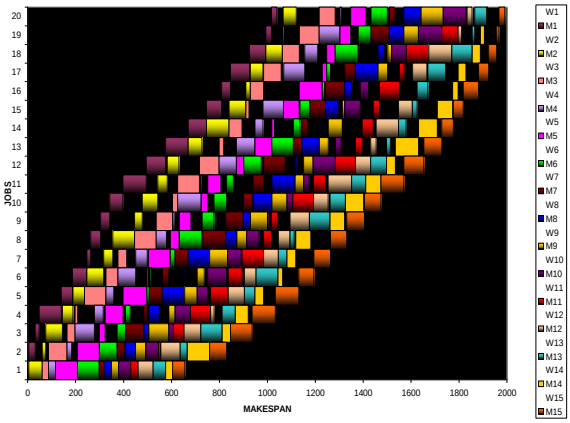


Fig. 2.1. GANTT CHART FOR REC 7

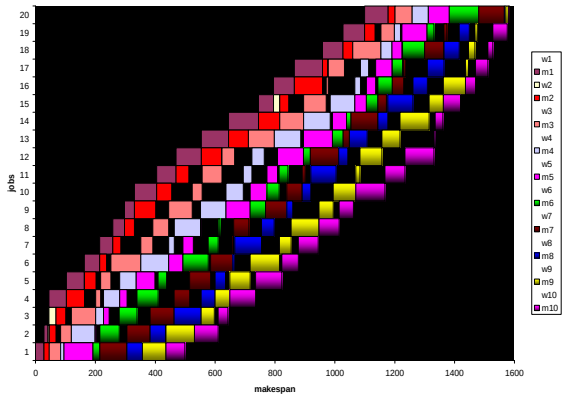


Fig. 2.2. GANTT CHART FOR REC15

Conclusion

The algorithms were developed and tested on benchmark problems of carlier and reeves. It was found that DMA produced accurate result but was able to solve small size problems of upto 17 jobs. The time consumption for bigger size problems was very big. The GA was able to solve bigger size problems of size 75 jobs and was able to give an result close to the optimum value. SA gave better result compared to GA, the performance of SA was enhanced by using the fast polynomial NEH algorithm. The best results were obtained for the hybrid algorithm of NEH+SA-GA. The hybrid algorithm gave the optimum results in some cases and for other problems the result obtained is very close to the optimum value. Thus it is concluded that the hybrid algorithm of NEH+SA –GA developed in this research is the most suitable algorithm for flow shop scheduling problems.

.

Problem	Size	Lower Bound	Computation Result					%Error	%Error	
			DMA	GA	SA	NEH+ SA	NEH+SA-GA	GA	NEH +SA	NEH+SA-GA
IPCAR 1	11 X 5	7038	7038	7038	7038	7038	7038	0	0	0
IPCAR 2	13 X 4	7166	7166	7166	7166	7166	7166	0	0	0
IPCAR 3	12 X 5	7312	7312	7318	7312	7312	7312	0.08	0	0

IPCAR 4	14 X 4	8003	8003	8003	8003	8003	8003	0	0	0
IPCAR 5	10 X 6	7720	7720	7720	7720	7720	7720	0	0	0
IPCAR 6	8 X 9	8505	8505	8505	8505	8505	8505	0	0	0
IPCAR 7	7 X 7	6590	6590	6590	6590	6590	6590	0	0	0
IPCAR 8	8 X 8	8366	8366	8366	8366	8366	8366	0	0	0
Rec 01	20 X 5	1242	***	1256	1251	1242	1242	1.12	0	0
Rec 03	20 X 5	1109	***	1130	1126	1113	1109	1.89	0.36	0
Rec 05	20 X 5	1242	***	1262	1259	1253	1246	1.61	0.88	0.32
Rec 07	20 X 10	1566	***	1584	1580	1574	1571	1.19	0.51	0.31
Rec 09	20 X 10	1537	***	1558	1556	1553	1541	1.36	1.04	0.26
Rec 11	20 X 10	1431	***	1467	1460	1452	1431	2.51	1.46	0
Rec 13	20 X 15	1930	***	1984	1976	1967	1935	2.79	1.91	0.25
Rec 15	20 X 15	1950	***	1993	1987	1972	1953	2.20	1.12	0.15
Rec 17	20 X 15	1902	***	1950	1942	1936	1902	2.52	1.78	0
Rec 19	30 X 10	2093	***	2150	2149	2141	2097	2.72	2.29	0
Rec 21	30 X 10	2017	***	2086	2071	2057	2024	3.91	1.98	0.34
Rec 23	30 X 10	2011	***	2088	2069	2063	2014	3.82	2.58	0.14
Rec 25	30 X 15	2513	***	2657	2657	2551	2027	5.7	1.51	0.55
Rec 27	30 X 15	2373	***	2458	2449	2436	2384	3.58	2.65	0.46
Rec 29	30 X 15	2287	***	2371	2363	2354	2303	5.07	2.92	0.69
Rec 31	50 X 10	3045	***	3136	3131	3104	3051	2.98	1.93	0.19
Rec 33	50 X 10	3114	***	3210	3197	3167	3123	3.08	1.7	0.28
Rec 35	50 X 10	3277	***	3325	3318	3308	3279	1.46	0.94	0.06
Rec 37	75 X 10	4890	***	5035	5028	5007	4982	2.96	2.39	1.88
Rec 39	75 X 10	5043	***	5201	5184	5141	5139	3.13	1.94	1.90
Rec 41	75 X 10	4910	***	5063	5027	4994	4962	3.11	1.71	1.05

Table – 1. Consolidated results for DMA, GA, SA, NEH+SA and NEH+SA-GA**.**

PROBLEM	**SIZE**	**K=0.001**	**K=0.04**	**K=0.4**	**K=0.8**	**K=1**
IPCAR 1	11 X 5	7415	7038	7281	7317	7729
IPCAR 2	13 X 4	7354	7166	7244	7288	7511
IPCAR 3	12 X 5	7482	7333	7401	7462	7531
IPCAR 4	14 X 4	8094	8007	8063	8099	8124
IPCAR 5	10 X 6	7843	7720	7816	7852	7892
IPCAR 6	8 X 9	8559	8506	8521	8593	8633
IPCAR 7	7 X 7	6645	6590	6612	6672	6690
IPCAR 8	8 X 8	8385	8366	8373	8397	8422
Rec 01	20 X 5	1346	1267	1328	1364	1384
Rec 03	20 X 5	1174	1124	1153	1183	1214

Rec 05	20 X 5	1298	1256	1271	1334	1381
Rec 07	20 X 10	1548	1522	1531	1575	1583
Rec 09	20 X 10	1583	1553	1572	1621	1643
Rec 11	20 X 10	1473	1452	1468	1485	1512
Rec 13	20 X 15	1991	1967	1982	2041	2164
Rec 15	20 X 15	2025	1972	2003	2074	2127
Rec 17	20 X 15	1958	1936	1940	1991	2029
Rec 19	30 X 10	2185	2141	2164	2231	2281
Rec 21	30 X 10	2098	2057	2071	2132	2167
Rec 23	30 X 10	2112	2063	2008	2056	2096
Rec 25	30 X 15	2583	2551	2562	2619	2658
Rec 27	30 X 15	2477	2436	2458	2489	2519
Rec 29	30 X 15	2372	2354	2360	2411	2437
Rec 31	50 X 10	3167	3104	3142	3192	3245
Rec 33	50 X 10	3217	3167	3199	3236	3276
Rec 35	50 X 10	3345	3308	3321	3362	3396
Rec 37	75 X 10	5075	5007	5036	5097	5135
Rec 39	75 X 10	5176	5141	5152	5189	5239
Rec 41	75 X 10	5029	4994	5015	5079	5086

Table 2 - Makespan for different values of K

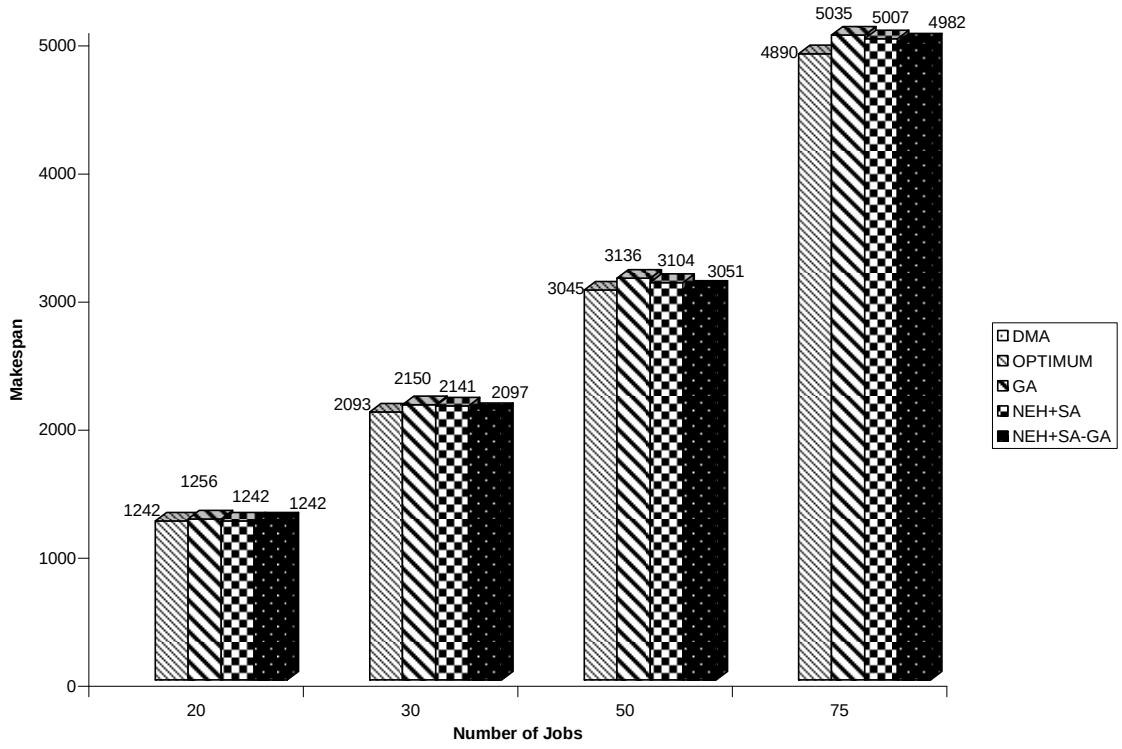


CHART 1 - Comparison of GA , NEH+SA and NEH+SA-GA Algorithms

REFRENCES

[1] Johnson, S. (1954). Optimal two- and three-stage production schedules with Setup times included. *Naval Research Logistics Quarterly,1*(1), 61–68.

[2] Palmer, D. (1965). Sequencing jobs through a multi-stage. Process in the minimum total time—a quick method of obtaining a near optimum. *Operations Research, 16*, 45–61.

[3] Campbell, H. R., & Smith, D. M. (1970). A heuristic algorithm for n-jobs m- machines sequencing problem. *Management Science,16*, 630–637.

[4] Nawaz, M., Enscore, J.,&Ham, I. (1983). A Heuristic algorithm for the M machine, n-job flow shop sequencing problem. OMEGA. *InternationalJournal of Management Science, 11*(1), 91–95.

[5] Rajendran, C. (1995). Theory and methodology heuristics for scheduling in flow shop with multiple objectives. *European Journal of Operational Research, 82*, 540–555.

[6] Temiz, I., & Erol, S. (2004). Fuzzy branch and bound algorithm for flow shop scheduling. *InternationalJournal of Intelligent Manufacturing,15*, 449–454.

[7] Koulamas, C. (1998). A new constructive heuristic for the flowshop scheduling problem. *European Journal of Operational Research Society, 105*, 66–71.

[8] Suliman, S. (2000). A two-phase heuristic approach to the permutation flow-shop scheduling problem. *International Journal of Production Economics, 64*, 143–152.

[9] Dipak Leha MIIE, Mukerjee SP (1998) Flow-shop scheduling using genetic algorithms for minimizing makespan. Indian Inst Ind Eng J 27:4–8

[10] Chen C-L, Vempati VS, Aljaber N (1995) Theory and methodology—an application of genetic algorithms for flow shop problems. Eur J Oper Res 80:389–396, 1995.

[11] T. Radha Ramanan · R. Sridharan · An artificial neural network based heuristic for flow shop scheduling problems J Intell Manuf DOI 10.1007/s10845-009-0287-5 Accepted: 24 June 2009 © Springer Science+Business Media, LLC 2009

[12] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of state calculation by east computing machines. J Chem Phys 21:1087–1091 doi:10.1063/1.1699114

[13] Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680 doi:10.1126/science.220.4598.671