Hybrid Flow Shop Scheduling using Improved Hybrid ACO Cuckoo Algorithm to Minimize Makespan

Ravianandan M
PG Scholar
Dept. of Manufacturing Engg.
College of Engg. Guindy

M. Omkumar, Ph.D.
Associate Professor
Dept. of Manufacturing Engg.
College of Engg. Guindy

ABSTRACT
Scheduling of jobs in manufacturing environment is often NP Hard[4].Multi stage Hybrid flow shop with a number of unequal parallel machines choices per stage makes it a further NP Hard to solve. Improved Hybrid ACO Cuckoo Algorithm proposed in this paper attempts to apply the algorithm to solve one such Hybrid flow shop problem. The performance of the algorithm was benchmarked against available Hybrid Algorithms to solve Hybrid flow shop. The outcome of the algorithm outperforms the performance of some of the Hybrid algorithm currently available.

General Terms
Flow shop Scheduling, Metahueristic Algorithms.

Keywords
Flexible Flow shop, Ant Colony Algorithm, Cuckoo Search, Makespan minimization.

1. INTRODUCTION
The use of meta-heuristic algorithms changed the face of scheduling in manufacturing industry. Increase in complexity of jobs being manufactured and variety of job being manufactured not only required computational expertise, but hybrid algorithms to find the best possible solution at the least time.

Flow shop scheduling is an important genre of NP-hard problems and requires extensive computational resources to perform scheduling. It is one of the most widely practiced types of scheduling. An extension of the flow shop is Hybrid flow shop (HFS) in which the jobs will have the flexibility to get processed by any of the identical machines at any of the flow stages.

The advent of evolutionary algorithms as meta-heuristic algorithms opened a new avenue of research and set out researchers to develop new algorithms and find the best for various problems. Genetic algorithm, particle swarm optimization algorithm, Ant colony algorithms, Fire Fly algorithms, Bat algorithm etc. are some of the earliest and most used for a wide variety of problems. Cuckoo Search (CS) algorithm has fared better than its earlier counterparts in many aspects like lesser number of parameters, faster and mature convergence (Xin-She Yang 2009).

This work intends to develop a hybrid ACO Cuckoo algorithm to solve a hybrid flow shop problem and demonstrate the effectiveness of the solution compared to the other hybrid algorithm.

2. LITERATURE STUDY
Omkumar et al[1] (2009) has performed research on static scheduling of multilevel assembly job shops using genetic algorithm. The scheduling had been done on multi-level job structures. Many benchmark dispatch rules such as SPT (shortest processing time), RAND (random selection of items), FIFO (first in first out), LF (latest finishing time), TWKR (total work remaining) have been used to schedule jobs on a static job shop and the schedules so generated had been used as initial population in a proposed genetic algorithm. The makespan thus obtained had been compared with many combinations of conventional dispatch rules and observed that GA gives an average reduction of 14.87% in makespan. However, the authors have limited their work to optimizing the makespan of the jobs through job scheduling and have not considered other important measures such as cost and tardiness.

Gandomi and Yang (2011) used Cuckoo Search algorithm to solve structural problems. He developed Cuckoo Search algorithm in combination with levy flights. The developed algorithm was first validated with Himmelblau's problem and found out that the proposed algorithm fared better than GA, PSO in terms of number of iterations and the optimal value found. The algorithm was then used with 13 structural optimization problems to find that they performed better than the state of the art techniques in the area. The author made a note that the speed of convergence of the algorithm did not depend on the parameters that are algorithm dependent (like probability of the host bird discovering the alien egg). The author also praised the algorithm for having much less number of parameters than GA or PSO.

A hybrid algorithm consisting of ACO and CSO algorithm was developed by Babukartik and Dhavachelvan (2012)[2] for scheduling jobs. He used the Cuckoo Search algorithm to search or explore the solution space more efficiently using levy flights. He calculated the time required for creation of the schedule using the algorithm and found that the time required increased with increase in number of jobs or tasks to be scheduled. The schedule was then run using a scheduler and found that the makespan was lesser in case of hybrid algorithm than that in ACO.

Kanagaraj et al[7] (2013) developed a hybrid algorithm consisting of GA and CSO for reliability-redundancy allocation problems. Doing so, they observed a better balance between exploration and exploitation.

FFSs are common in industries of various realms like electronics, textile and paper industries (Wang and Liu 2013)[12]. A flow shop may be made flexible due to reasons like the management may not be ready to dispose the old machines even after purchase of a new one or if certain processes in the flow line takes substantially more time than the rest, use of several identical machines at that stage may be reasonable so as to make the flow speed of the jobs uniform and hence reduce piling up of jobs in between machines, if that is undesirable. Scheduling of jobs in FFSs of the smallest size

possible with same process times for a job on identical machines itself has been proven to be NP-hard (Gupta et al. 1997). So, any FFS with a job having different process times on identical machines (in terms of the processes they can carry out) should be NP-hard too.

Marichelvan, Prabaharan & Yang[9] (2014) improved Cuckoo search algorithm for hybrid flow shop by using NEH heuristic prior to the cuckoo search algorithm. The details of the hybrid flow shop were obtained from a furniture industry and the performance of the improved cuckoo search algorithm was compared with that of its peers. It not only showed improvement in terms of the average of minimum makespan obtained, but also in terms of the best makespan obtained.

Li-Ning Xing ,Ying-Wu Chen ,Peng Wang, Qing-Song Zhao,Jian Xiong (2009)[11] used Knowledge Based Ant colony Optimization (KBACO) to solve Hybrid flow shop Scheduling Optimization. Li-Ning Xing et al combined Metaheuristic and knowledge model to arrive at knowledge based KBACO model. Knowledge model learns some available knowledge from the optimization of ACO and then applies the existing knowledge to guide the current heuristic searching. A knowledge of Job Machine probability matrix is developed and Ant algorithm is influenced by the knowledge model to choose a appropriate machine to process the job.

Ruben Ruiz(2009) et al[10] , have reviewed and analyzed more than 200 papers dealing with the hybrid flow shop (HFS) or related variants. He has classified all the papers according to many parameters, including problem variant studied, constraints, objective functions and employed methodologies.

The author's identified only very small percentage of papers is dedicated to the solution of problems with real world motivated functions. They found large number of papers with minimization of makespan as objective function though a lot other exist. According to Ruiz, Production scheduling problems are multi-objective (MO) by nature, which means that several criteria, in conflict with each other, have to be considered at a time.

From the literatures, it is understood that Cuckoo and Ant colony algorithm are two powerful tools for solving complex problems in two different perspectives, one by exploiting and the other by exploring. A hybrid flow shop is a non-linear NP-hard problem and requires good balance of exploration and exploitation in solving. No much research work has been found using a powerful algorithm which has both elitist and yet explorative capability for a complex non-linear NP-hard problem. This work is an attempt to bridge the gap, employing a hybrid ACO and Cuckoo search in a hybrid flow shop environment.

3. PROBLEM DEFINITION

3.1 Hybrid Flow Shop – Metal Spinning Process Case Study

Metal spinning process with multiple stages and multiple machines per stage is shown in Figure 1.0

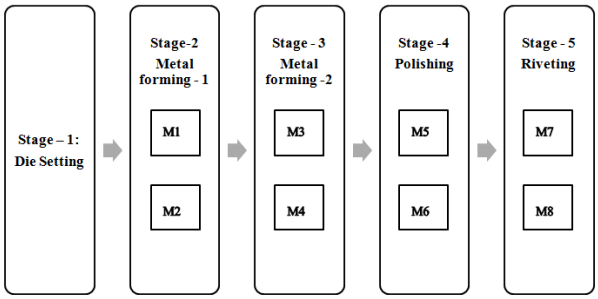

Fig 1: Metal Spinning Process Flow – A typical flow shop configuration.

Each job processed in metal spinning may go through all of the process stages or only a few of them. Not all stages of operation are applicable for all jobs that are being processed

3.2 Field Data Collection

Data were collected from aluminum metal spinning industry and the average process time for each job is presented below in Table 1.0

Table 1. Average Job processing time for jobs

Job Processing time (in seconds)									
Job Number	**Stage 1 - die Setting**	**Stage 2 - Metal Forming 1**		**Stage 3 - Metal forming 2**		**Stage 4 - Polishing**		**Stage 5 - Riveting**	
		M1	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**
1	600	52	50	44	45	54	56	0	0
2	580	63	60	46	48	53	53	0	0
3	560	53	52	0	0	82	88	0	0
4	600	54	52	0	0	64	64	15	12
5	640	54	52	0	0	64	64	15	12
6	600	30	31	0	0	9	8	0	0
7	620	37	37	42	43	27	26	0	0
8	600	36	36	45	43	43	45	0	0
9	611	63	64	51	50	49	49	0	0
10	588	54	54	52	57	50	50	0	0

3.3 Constraints

In this work, no no-wait constraint is assumed and it is assumed that enough buffer space is made available just in case. The following assumptions are made:

1. Each machine can perform only one operation at a time
2. No pre-emption is possible
3. Jobs cannot be re-sequenced after the first stage
4. Process time includes set-up time as well

5. A job can be processed only once at a stage
6. A job can move only stage by stage
7. Every job can be processed by not more than a single machine at a time

A mathematical model of the proposed scheduling operation is given below and the notations are described along with.

J_i *where* $1 \le i \le n$ = Time at which the previous process was completed on a job.

M_k *where* $1 \le k \le m$ = Time at which the previous process was completed on a machine

$P_{i,j}$ = Process time for job *i* on machine *j*

Initially, $M_j = 0$ and $J_i = 0$

J_{i1} = Increment in machine and job completion times for a job *i* in the first stage

$JC_{i1} = P_{ij} \mid (M_j + P_{i-1j})$ is the minimum for any *j*, where $j = 1, 2 \ldots mq1$ is the selected machine in first stage

$$J_i = M_j + JC_{i1}$$

$$M_j = M_j + JC_{i1}$$

$JC_{i2,3,p}$ = Increment in machine and job completion times for a job in the second or third stage

$JC_{i2,3,p} = P_{i,j} \mid (max(M_j, J_i) + P_{i,j})$ is the minimum for any *j*, where $j = mq1+1, \ldots, mq2$ is the selected machine in second stage or any such stage.

$$J_i = max(J_i, M_j) + JC_{i2,3,..p}$$

$$M_j = max(J_i, M_j) + JC_{i2,3...p}$$

3.4 Makespan Calculation

Objective Function is to minimize make span i.e time between start of first job and completion of last job.

$$Cmax = Max(C_{i-1j}, C_{j-1i}) + P_{ij}$$.

Where

C_{i-1j}, C_{j-1i} = Completion time of previous operation.

Pij = Processing time of next Operation.

Subject to machine availability in each stage to process the job.

4. HYBRID ANT CUCKOO ALGORITHM

Hybrid Ant Cuckoo algorithm uses the power of Ant and cuckoo to both explore and exploit the search space i.e. job sequence to arrive at best possible solution i.e. minimized makespan.

4.1 Methodology

The below flow chart in figure 2.0 shows the implementation methodology of the Ant colony algorithm and cuckoo algorithm in hybrid flow shop environment.

Java 6.0 with Netbeans IDE 8.0 is used as development environment for hybrid algorithm implementation.

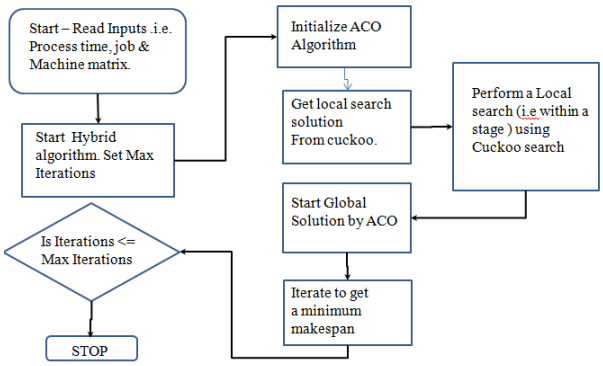

Fig 2: Hybrid ACO Cuckoo algorithm implementation methodology.

The hybrid algorithm is developed using using Java 6.0 Programming language.

To arrive at optimized parameters for ant colony and cuckoo search, a number of iterations were run with normally distributed random numbers with mean 100 and variance 10.

Table 2. Random Data Used for Analysis

Machining Time (in Seconds)								
	Stage 1		Stage 2		Stage 3		Stage 4	
	M1	M2	M3	M4	M5	M6	M7	M8
Job1	82	74	106	97	82	74	106	97
Job2	102	89	104	108	102	89	104	108
Job3	114	117	93	104	114	117	93	104
Job4	92	96	80	101	92	96	80	101
Job5	93	103	95	92	93	103	95	92
Job6	82	74	106	97	82	74	106	97
Job7	102	89	104	108	102	89	104	108
Job8	114	117	93	104	114	117	93	104
Job9	92	96	80	101	92	96	80	101
Job10	93	103	95	92	93	103	95	92

Different trials were performed with data available in the table 6.2 and it found the following values provided near optimal solution at faster rate.

The table shows the optimized parameters for Ant and cuckoo algorithms.

Table 3. Optimized parameters

Algorithm	Parameter	Value
Ant colony	Initial Pheromone Trial	1

	Alpha	1
	Beta	5
	Evaporation	0.1
	Pheromone deposit Coefficient	300
	Probability	0.6
	Nest size	20
	Probability of eggs	0.5
Cuckoo Search	Number of iterations	1000

The end condition i.e. number of iterations is fixed best on the solution convergence value.

4.1.1 Role of Cuckoo Search Algorithm

Hybrid flow shop problem involves two basic steps

1. Assignment of machines to jobs based on LPT rule.
2. Job Sequencing of job process time matrix.

Cuckoo search algorithm is effectively used in this algorithm to assign jobs to machines. Cuckoo fitness function is based on LPT rule. Cuckoo finds the best solution from a number of solutions from given set of available solutions.

Cuckoo role ends when the final job-machine matrix is arrived at and also the job process time matrix corresponding to job machine matrix.

4.1.2 Role of Ant colony Algorithm

Ant colony algorithm takes the job process time matrix as input and iterates the job sequence based on pheromone distribution for each ant tour trials. The best solution is retained and further trials are conducted until the end condition is met.

The ant colony algorithm provides the best job sequence corresponding to minimum makespan function.

5. RESULTS AND CONCLUSIONS

Table below shows the best minimum makespan obtained for each problem configuration.

Table 4. Results Comparison

no of jobs	no of stages	machines per stage	algorithm	sequence	makespan
10	5	2	FIFO	1 2 3 4 5 6 7 8 9 10	7141
10	5	2	ACO - CS	9 1 6 4 10 3 2 8 7 5	7039

6. CONCLUSIONS

It can be concluded that the hybrid algorithm performed better than its peers at all problem configurations in terms of the minimum makespan. When the number of jobs was small, the average of makespan obtained using all the algorithms were more or less equal.

The above conclusion is true also when the number of stages was less. The reason could be that when the number of stages is less, all the machines would be occupied mostly. The hybrid algorithm was seldom outperformed by others in terms of the best makespan obtained. But this could be due to sheer chances.

The type of algorithm used is significant with a high confidence level at all possible number of jobs, machines and sections, and the hybrid algorithm is the best.

7. ACKNOWLEDGMENTS

I express my sincere thanks to Dr.M.omkumar for his valuable feedback and suggestions.

8. REFERENCES

[1] Omkumar, M, Shahabudeen, P, Gughan, S & Azad, 2009, 'GA based static scheduling of multilevel assembly job shops', International journal for Operational Research.

[2] Babukartik, R G & Dhavachelvan P.,2012, 'Hybrid Algorithm using the advantage of ACO and Cuckoo Search for Job Scheduling', International Journal of Information Technology Convergence and Services, vol.2, No.4, pp. 1-10

[3] Ebrahimi.M, Fatemi S.M.T., Karimi B., 2014, 'Hybrid Flow Shop Scheduling with sequence dependent family setup time and uncertain due dates', Journal of Applied Mathematical Modeling, Vol. 38 (2014), pp. 2490-2504.

[4] Gupta, J N D, Hariri *et al.*, 1997, 'Scheduling a two-stage hybrid flow shop with parallel machines at the first stage', Annals of Operations Research, vol. 69, pp. 171 – 191.

[5] Imma Ribas, Rainer Leisten, Jose M. Framinan , 2010, 'Review and Classification of hybrid flow shop scheduling problems from a production system and a solution procedure perspective', Journal of Computers & Operations Research, Vol. 37 (2010), pp. 1439-1454.

[6] Kenneth R. Baker ,2013, 'Minimizing earliness and tardiness costs in stochastic scheduling', European journal of operation research,Vol. 236 (2014),pp. 445–452.

[7] Kanagaraj, G, Ponnambalam, S G & Jawahar, N 2013, 'A hybrid Cuckoo Search and genetic algorithm for reliability– redundancy allocation problems', Computers & Industrial Engineering, pp. 1-10

[8] Li-Ning Xing,Ying-WuChen,Peng Wang,Qing-Song Zhao,Jian Xiong, 2010, 'A Knowledge Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems', Journal of Applied soft Computing, vol.10, pp.888-896..

[9] Marichelvam, M, Prabaharan, T, & Yang, X S 2014, 'Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan', vol. 19, Applied Soft Computing, pp. 93-101.

[10] Ruben Ruiz , Jose Antonio,Vazquez-Rodriguez 2010,'The Hybrid flow shop Scheduling problem', European Journal of Operation Research, Vol. 205 (2010), pp. 1-18.

[11] RuiXua,HuapingChena,XuepingLi(2013),'A bi-objective scheduling problem on batch machines via a Pareto-based ant colony system', Journal of production economics, Vo1.134(2013),pp.371-386.

[12] Wang, S & Liu, M 2013, 'A genetic algorithm for two-stage no-wait hybrid flow shop scheduling problem', Computers & Operations Research, vol. 40, pp. 1064-1075.