# No-wait Flowshop Scheduling Using Genetic Algorithm

Imran Ali Chaudhry *Member, IAENG*, Sultan Mahmood

***Abstract*— No-wait flowshop scheduling is a constrained flow shop scheduling problem that exists widely in manufacturing systems. This paper considers minimization of makespan (total completion time) for *n* number of jobs to be processed on *m* machines using a general purpose spreadsheet based genetic algorithm (GA). The proposed approach solution is compared with already published problems in the literature. The proposed approach produces optimal solution for all cases. Additionally it is also shown that any objective function can be minimised using the same model without changing the logic of the GA routine.**

***Index Terms*—Genetic Algorithms, No-wait, Makespan, Scheduling, Flowshop.**

## I. Introduction

Scheduling of manufacturing systems is an important aspect for any enterprise to maintain a competitive position in fast-changing markets, hence it is important to develop effective, efficient advanced manufacturing & scheduling technologies and approaches The goal of scheduling is to maximize (or minimize) different criteria of a facility such as makespan, flow time, total tardiness etc.

The flowshop scheduling problem is an important scheduling problem which been extensively studied since it was proposed in 1954 by Johnson [1]. A detailed review of flowshop scheduling research is given by Stafford [2].

In this paper we present a spreadsheet based GA approach to minimize the makespan for a no-wait flowshop scheduling problem.

The rest of the paper is organised as follows: Section II reviews the no-wait flowshop scheduling problem (FSSP), followed by problem and assumptions in Section III. Section IV gives the GA details as implemented in this paper. Experimental results are given in Section 4. The last section of the paper presents conclusions.

## II. LITERATURE REVIEW

No-wait flow shop scheduling problems have been most commonly studied with two optimization (minimization) criteria: total flow time and makespan. According to the research work by Garey and Johnson [3], the no-wait FSSP is NP-hard even for the two machine case. Similarly Rock [4] proved that no-wait FSSP with makespan criterion is NP-hard. Due to its significance in various applications: both theory and industry, the no-wait FSSP has attracted the attention by many researchers.

In many flow shops, there may be a constraint that once the processing of a job begins, subsequent processes of that job must be carried out without any delay. If necessary, the start of job processing can be delayed on the first machine so that the job need not wait for processing on the subsequent machines. Such a flow shop can be termed a 'constrained flow shop' or 'no-wait flow shop'. Examples of this type of shop occur in hot metal rolling industries. In this process, a heated slab is rolled through a series of rolling mills in which the slab thickness, or gauge, is successively reduced to some final specification. In this kind of processing, the slab must pass directly from one rolling mill to the next rolling mill in the sequence without any waiting, as such waiting would result in cooling the slab to such a temperature that the metal could not be rolled easily. Some other examples of this situation arise in chemical or metal processing industries. The no-wait situation also occurs in a production environment where there is a lack of storage between the machines. This is the case in JIT production lines where the flow of jobs is continuous with no in process inventory.

Hall and Sriskandarajah [5] give a detailed survey of the research and applications of no-wait flow shop scheduling problem.

Makespan criterion can be defined as completion time at which all jobs complete processing or equivalently as maximum completion time of jobs. A review of flowshop scheduling with makespan criterion has been given be Hijazi and Shaghafian [6]. The no-wait flowshop problem with makespan criterion has been considered by [7]-[23].

## III. PROBLEM AND ASSUMPTIONS

The no-wait flow shop scheduling problem is described as follows [24]: Each of $n$ jobs from set $J=\{1,2,....., n\}$ will be sequenced through $m$ machines ($k=1,2,..., m$). Job $j \in J$ has a sequence of m operations ($o_{j1}, o_{j2},...., o_{jm}$). Operation $o_{jk}$ corresponds to the processing of job $j$ on machine $k$ during an un-interrupted process time of $p(j, k)$. At any given time, each machine can process at most one job and each job can be processed on at most one machine. In order to follow the no-wait restriction, the completion time of the operation $o_{jk}$ must be equal to the earliest start time of the operation $o_{j,k+1}$ for $k=1,2,....., m-1$. In other words, there must be no waiting time between processing of any consecutive operations of each of $n$ jobs. The problem is to find a schedule such that the processing order of jobs is the same on each machine and the maximum completion time so called makespan is minimized.

Manuscript received January 04, 2011.
I. A. Chaudhry (imran_chaudhry@yahoo.com) and S. Mahmood (sultanrandhawa@hotmail.com) are with National University of Sciences & Technology, Islamabad.

Some commonly used assumptions in the deterministic no-wait flow shop problem are (i) known and deterministic process times on all machines, (ii) no pre-emption allowed, and (iii) jobs follow the same technological order of machines without any interruption either on or between them for each job.

## IV. GENETIC ALGORITHMS

Genetic algorithm is a bio-inspired method that is applicable in continuous as well as discrete optimization and has shown excellent performance in various scheduling problems. Unlike other metaheuristics, in a GA a set of solutions is kept during the optimization process. This set is called population and each individual in the set represents an solution to the problem. The population is then evaluated and all individuals are assigned a fitness value with the idea that a higher fitness value should be assigned to better individuals. As in natural evaluation of species, the population undergoes a series of operations. Individuals in the population evolve during generations until some stopping criterion is met. In a particular generation, firstly individuals from the current population are selected by a selection mechanism. Thus, fitter individuals have a more chance of being selected. Selected individuals then undergo crossover process where they mate, thus producing new individuals. Crossover operator generates better solutions that are biased by the good solutions represented by the parents. In order to create diversity in population, the newly produced individuals mutate in an effort to include a possibility of visiting any possible solution in the search space. After each generation, new individuals are recreated and the new population is again evaluated. The whole process is then repeated.

A detailed introduction to Gas is given in Goldberg [25]. The earliest application of GA has been reported by Davis [26]. For a recent review of GA application in scheduling is given in Chaudhry and Drake [27] and Chaudhry [28].

In this paper we use a commercial GA package EVOLVER™ [29], which functions as an add-in to Microsoft Excel™. The model for no-wait FSSP is built in Microsoft Excel using the spreadsheet’s built in functions. Within the familiar spreadsheet environment, Evolver generates a number of trial solutions and uses genetic algorithms to continually improve results of each trial. Each possible solution then becomes an independent "organism" that can "breed" with other organisms. The spreadsheet model acts as an environment for this population of solutions, determining which are "fit" enough to survive based on their results. The objective function, variables, and the constraints are readily specified by highlighting the corresponding spreadsheet cells. Fig. 1 illustrates the Evolver-spreadsheet integration.

The advantage of the proposed method is that the program runs in the background freeing the user to work in the foreground. Moreover, the familiar layout of the spreadsheet software makes it easier for the user to use the software and the presentation of data in tables makes it easier for the user to carry out what-if analysis.

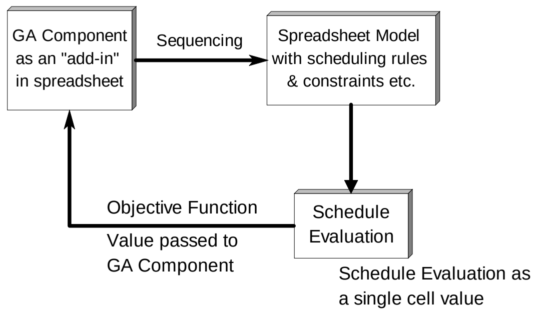
GA Component as an "add-in" in spreadsheet
Sequencing
Spreadsheet Model with scheduling rules & constraints etc.
Objective Function
Schedule Evaluation
Value passed to GA Component
Schedule Evaluation as a single cell value

Fig. 1: Evolver- spreadsheet Integration

### A. *Chromosome Representation*

For no-wait FSSP, we use permutation representation where each job represents each gene. For example, in a flow shop with five jobs i.e. A-B-C-D-E, one chromosome according to permutation representation could be A-C-B-E-D, while another could be E-D-C-B-A.

### B. *Reproduction/Selection*

Evolver uses a steady state approach. The steady-state method selects two chromosomes according to the selection procedure, performing crossover to obtain one child, perhaps applying mutation as well, and installs the result back into that population; the least fit of the population is destroyed.

As far as parent selection is concerned, in Evolver, parents are chosen using a rank-based mechanism. In this method, rather than absolute fitness, the selection probabilities are based on a chromosome’s relative rank or position in the population. In rank-based fitness, population is sorted according to the objective values. The fitness assigned to each individual depends only on its position in the individual rank and not on the actual objective value.

### C. *Crossover Operator*

The crossover operator is the most important in GA. Traditionally, crossover is a process yielding recombination of bit strings via an exchange of segments between pairs of chromosomes. For the no-wait FSSP, we needed to handle permutation representation; the “Order Solving Method” of Evolver was used. This method applies order crossover operator [26]. An offspring is built by choosing a subsequence from one parent and preserving the relative order of jobs from the other. . For example, the two parents

[P1] = ( 5 7 | 2 1 9 3 | 6 8 4 ) and
[P2] = ( 8 6 | 3 7 1 5 | 4 2 9 )

First, the segments between cut points are copied into offspring

[O1] = ( _ _ 2 1 9 3 _ _ _ ) and
[O2] = ( _ _ 3 7 1 5 _ _ _ )

Next, starting from the second cut point of [P1] parent, the jobs from the other parent are copied, except which are already in offspring [O1], we get 4-8-6-7-5. This sequence is placed in the first offspring.

[O1] = (7 5 2 1 9 3 4 8 6).

Similarly we get the other offspring: [O2] = (2 9 3 7 1 5 6 8 4).

### D. Mutation Operator

With crossover, the search is constrained to alleles which exist in the initial population. The mutation operator can overcome this by simply randomly selecting any bit position in a string and changing it. This is useful since crossover may not be able to produce new alleles if they do not appear in the initial generation. To preserve all the original values, the "Order Solving Method" performs order-based mutation. In this type of mutation, two positions are selected randomly, and two characters (genes) in these positions are interchanged. For example, in a parent [1 2 3 4 5 6 7 8 9], with selected positions 2 and 5, the resultant child would be [1 5 3 4 2 6 7 8 9]. The number of swaps increase or decrease proportionately to the increase and decrease of the mutation rate setting (from 0 to 1).

## V. EXPERIMENTAL RESULTS

### A. Implementation Details

Consider the $5/3/F/C_{max}$ example given in Table 1.

Table 1: Job Data for example no-wait FSSP

| | Processing Time on Machine | | |
|---|---|---|---|
| **Job** | ***1*** | ***2*** | ***3*** |
| 1 | 3 | 2 | 4 |
| 2 | 4 | 5 | 3 |
| 3 | 1 | 4 | 5 |
| 4 | 1 | 3 | 2 |
| 5 | 4 | 3 | 7 |

A Gantt chart for the sequence of jobs 5-3-2-1-4 produced with wait times between the jobs is shown in Figure 1.

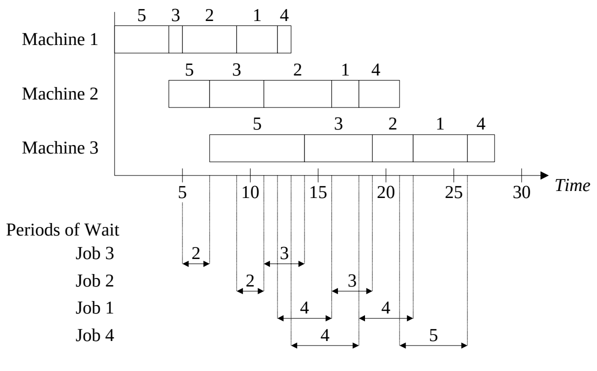

Fig. 2: Gantt chart showing job sequence 5-3-2-1-4 and the periods of wait for the jobs

The 'no-wait' spreadsheet model works in two stages. In the first stage the delay in the start of the job $p$ after $q$ is calculated. In the second stage the job is delayed by that many units as calculated in stage one to produce a valid no-wait schedule. The delay of starting job $q$ after $p$ is calculated by the equation given by Reddi and Ramamoorthy [8]. Let $D(p, q)$ be the minimum delay time between the completion of job $J_p$ and the initiation of $J_q$, then $D(p, q)$ is given by:

$$D(p,q) = \max(t_{p,2}-t_{q,1},\ t_{p,2}+t_{p,3}-(t_{q,1}+t_{q,2}),\ldots\ldots,\ t_{p,2}+t_{p,3}+t_{p,4}+\ldots\ldots+t_{p,m}-(t_{q,1}+t_{q,2}+\ldots\ldots+t_{q,m-1}),0)$$

$$= \max_k\left(\sum_{i=2}^{k} t_{p,i} - \sum_{i=1}^{k-1} t_{q,i}, 0\right),$$

$$2 \le k \le m$$

where

$t_{p,i}$ = process time for job $p$ on machine $i$

$m$ = number of machines

After delaying the start of job $q$ after job $p$ by duration $D(p, q)$, the schedule in Figure 1 would be a no-wait schedule and is as given in Figure 2.

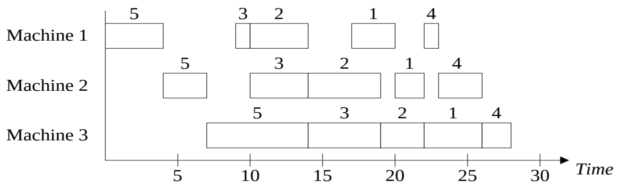

Fig. 3: Gantt chart showing job sequence 5-3-2-1-4 for no intermediate storage

### B. Computational Analysis

In order to check the effectiveness of the proposed spreadsheet based GA approach we took problems from already published literature. The problems have been simulated on a Dual Core 2.88 GHz computer having 512 MB RAM. By conducting repeated tests, we found that the best values to be set for the number of population, the crossover rate, and the mutation rate are 60, 0.65, and 0.09, respectively. Therefore, for each of the run, same set of parameter setting as described previously have been used, which correspond to 1 min 20 s on a Dual Core 2.88 GHz computer having 512 MB RAM.

Five example problems with different objective function were taken from the literature to demonstrate this approach. A summary of the results is given in Table 2.

| **Prob** | **Reference** | **Objective Function** | **Approach** | **Result** | **GA Result** |
|---|---|---|---|---|---|
| 1 | Wismer [9] | Makespan | Branch and Bound | 42 | 42 |
| 2 | Van Deman & Baker[10] | Average flow time | Branch and Bound | 28 | 28 |
| 3 | Szwarc [13] | Makespan | Gilmore-Gomory Algorithm [30] | 88 | 88 |
| 4 | Rajendran & Chaudhri [14] | Total flow time | Heuristic approach | 503 | 501[1] |
| 5 | Rajendran [16] | Makespan | Heuristic approach | 20 | 20 |

[1] This is shown to be an optimal solution given by Rajendran and Chaudhri [14].

## VI. Conclusion

In this paper, we considered the problem of scheduling jobs in a no-wait flowshop. The problem is known to be NP-hard. A spreadsheet based general purpose GA solution methodology was presented. The spreadsheet GA implementation has been found to be easy to implement catering for the peculiarities of any environment. Moreover, the spreadsheet environment makes it very suitable to carry out what if analysis. The spreadsheet model can be easily customized to include additional jobs, machines or workers without actually changing the logic of the GA routine thus making it a general purpose scheduling approach. We also demonstrate that any objective function can be used without changing the logic of the GA routine.

The proposed approach was able to find optimal solution for all the problems with different objective function thus demonstrating the robustness of the proposed approach.

## References

[1] S. M. Johnson, “Optimal two-and three-stage production schedules with setup times included”, *Naval Research Logistics Quarterly*, vol. 1, 1954, pp. 61-68.

[2] J. N. D. Gupta, E. F. Stafford Jr., “Flowshop scheduling research after five decades”, *European Journal of Operational Research*, vol. 169(3), 2006, pp. 699-711.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.

[4] H. Rock, “The three-machine no-wait flowshop problem is NP-complete”, *Journal of the Association for Computing Machinery*, vol. 31(2), 1984, pp. 336-345.

[5] N. G. Hall, C. Sriskandarajah, “A survey of machine scheduling problems with blocking and no-wait in process”, *Operations Research*, vol. 44(3), 1996, pp. 510-525.

[6] S. R. Hejazi, S. Saghafian, “Flowshop-scheduling problems with makespan criterion: a review”, International Journal of Production Research, vol. 43(14), 2005, pp. 2895-2929.

[7] J. Piehler, Ein Beitrag Zum Reinhenfolgeproblem, *Unternehmensforschung*, vol. 4, 1960, pp. 138-142.

[8] S.S. Reddi, C.V. Ramamoorthy, “On the flowshop sequencing problem with no-wait in process”, *Operational Research Quarterly*, vol. 23, 1972, pp. 323–331.

[9] D. A. Wismer, “Solution of the flowshop-scheduling problem with no intermediate queues”, *Operations Research*, vol. 20(3), 1972, pp. 689-697.

[10] J. M. Van Deman, K. R. Baker, “Minimizing mean flowtime in the flow shop with no intermediate queues”, *AIIE Transactions*, vol. 6(1), 1974, pp. 28-34.

[11] M. C. Bonney, S. W. Gundry, “Solution to the constrained flowshop sequencing problem”, *Operations Research Quarterly*, vol. 24, 1976, pp. 869-883.

[12] J. R. King, A. S. Spachis, “Heuristics for flow-shop scheduling”. *International Journal of Production Research*, vol. 18, 1980, pp. 345-357.

[13] W. Szwarc, “Solvable cases of the flow-shop problem without interruptions in job processing”, *Naval Research Logistics Quarterly*, vol. 30, 1983, pp. 179-183.

[14] C. Rajendran, D. Chaudhuri, “Heuristic for continuous flow-shop problem”, *Naval Research Logistics*, vol. 37, 1990, pp. 695-705.

[15] R. Gangadharan, C. Rajendran, “Heuristic algorithms for scheduling in the no-wait flowshop”, *International Journal of Production Economics*, vol. 32, 1993, pp. 285-290.

[16] C. Rajendran, “A no-wait flowshop scheduling heuristic to minimize makespan”, *Journal of the Operational Research Society*, vol. 45(4), 1994, pp. 472-478.

[17] T. Aldowaisan, A. Allahverdi, “New heuristics for no-wait flowshops to minimize makespan”, *Computers & Operations Research*, vol. 30, 2003, pp. 1219-1231.

[18] J. Grabowski, J. Pempera, “Some local search algorithms for no-wait flow-shop problem with makespan criterion”, *Computers & Operations Research*, vol. 32, 2005, pp. 2197-2212.

[19] P.J. Kalczynski, J. Kamburowski, “On no-wait and no-idle flow shops with makespan criterion”, *European Journal of Operational Research*, vol. 178, 2007, pp. 677-685.

[20] J. M. Framinana, M. S. Nagano, “Evaluating the performance for makespan minimisation in no-wait flowshop sequencing, *Journal of Materials Processing Technology*, vol. 197, 2008, pp. 1-9.

[21] Q-K. Pan, L. Wang, M. F. Tasgetiren, B-H. Zhao, “A hybrid discrete particle swarm optimization algorithm for the no-wait flow shop scheduling problem with makespan criterion”, *International Journal of Advanced Manufacturing Technology*, vol. 38, 2008, pp. 337-347.

[22] Q-K. Pan, L. Wang, B-H. Zhao, “An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion”, *International Journal of Advanced Manufacturing Technology*, vol. 38, 2008, pp. 778-786.

[23] D. Laha, U. K. Chakraborty, “A constructive heuristic for minimizing makespan in no-wait flow shop scheduling”, *International Journal of Advanced Manufacturing Technology*, vol. 41, 2009, pp. 97-109.

[24] Q-K. Pan, L. Wang, B. Qian, “A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems”, *Computers & Operations Research*, vol. 36(8), 2009, pp. 2498-2511.

[25] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, 1989, Addison-Wesley, Boston.

[26] L. Davis, *Handbook of Genetic Algorithms*, 1991, New York, Van Nostrand Reinhold

[27] I. A. Chaudhry and P. R. Drake, “Minimizing total tardiness for the machine scheduling and worker assignment problems in identical parallel machines using genetic algorithms”, *International Journal of Advanced Manufacturing Technology*, vol. 42, 2008, pp. 581-594.

[28] I. A. Chaudhry, “Minimizing flow time for the worker assignment problem in identical parallel machine models using GA”, *International Journal of Advanced Manufacturing Technology*, 2009, vol. 48(5-8), 2010, pp. 747-760.

[29] Palisade Corporation, Evolver: the genetic algorithm super solver. 1998, New York, USA.

[30] P. C. Gilmore, R. E. Gomory, “Sequencing a one-state variable machine : A solvable case for the travelling salesman problem”, *Operations Research*, vol. 12, 1964, pp. 655-679.