Fuzzy Cell Genetic Algorithm Approach for Flexible Flow-Line Scheduling Model

Arash Nasrolahi Shirazi[*], Meghan Steinhaus[†], Matthew Agostinelli[*], Manbir Sodhi[*]
[*] Department of Mechanical, Industrial, and Systems Engineering
University of Rhode Island
Kingston, RI 02881
arashshirazi@uri.edu
[†]Department of Mathematics, US Coast Guard Academy,
New London, CT 06320

***Abstract*—This paper focuses on makespan minimization for the flow line scheduling problem using a Fuzzy Cell Genetic Algorithm (FCGA). Real world applications of this problem are commonly found in printing and electronic circuit board manufacturing industries. A generalized integer programming (IP) model for this problem is proposed. The Fuzzy Cell Genetic Algorithm (FCGA) is proposed to solve the IP model, which has been proven to be NP-hard. Sample problems are generated with known good solutions to evaluate the effectiveness of the FCGA approach. The FCGA matches the performance of the IP model for small sized problem instances and it is proven to be effective for larger problem instances.**

I. Introduction

The flow line sequencing problem is defined by a set of jobs, and each job follows the same route from one machine to another throughout a manufacturing system. The sequence problem is determine the position of each job relative to other jobs for processing. The general form considered here has been derived from traditional manufacturing systems where each stage is comprised of one machine that all jobs must pass through. The expanded version of flow line sequencing is this flexible flow line sequencing problem, where each stage consists of more than one machine in parallel for all stages of operation to allow for concurrent processing in each stage. The flexible flow line is synonymous with a hybrid flow shop except that only one stage needs to feature parallel machines in a hybrid flow shop where the flexible flow shop considers parallel machines in all stages. Additionally, hybrid flow shops allow jobs to skip stages where flexible flow shops require jobs to visit all stages in order[1, 2].

Flexible flow lines have been used to model many real world environments including the production of circuit boards, automobiles, paper, textiles, and concrete[3–8]. Formally, flexible flow line systems can be defined by a set of n jobs to be processed in m stages with p number of parallel machines in each stage. Generally, flexible flow line problems have the following features in common:

1) The number of stages to perform the jobs must be greater than one.
2) The total number of stages must be less than the total number of machines.
3) All jobs must be processed in every stage but it is optional which machine processes the job in that stage.
4) Each job has a processing time represented by $Pt_{t,s}$ where t is the index of job to process and s is the s^{th} stage

In the general form of the flexible flow line problem, all machines in each stage are ready to perform the jobs at time zero, the parallel machines in each stage are identical, each machine can perform just one job at a time, and any job can be processed by any machine in a given stage. Setup time is negligible in the general form and buffers are infinite between stages. Figure 1 provides a simplified graphical form of the flexible flow line. The jobs are indexed by t, the stages notated by s and each machine notated by k in each work center to process the jobs.

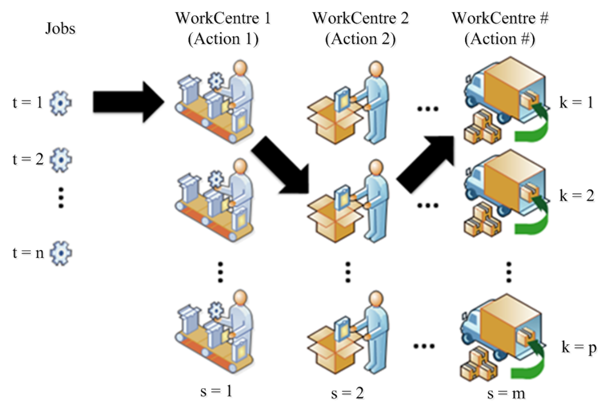

Fig. 1. Graphical representation of the problem design which j, s and p are index representative of jobs, stages and machines respectively.

Generally, the objective of flexible flow line optimization is to minimize the makespan as a scheduling criteria. The makespan is defined as the time between the start of the first job in the system and the completion of the last job. Minimizing the makespan reduces the maximum completion time where all jobs are considered. This means each job has a completion time in the last stage and the maximum of all completion times is the makespan for the system. Traditionally, makespan minimization has been the focus of flexible flow line literature. However, some researchers have used maximum capacity volume and total flow time as objectives [9].

Several approaches have been taken to solve the traditional flow shop, parallel process shops, and hybrid flow shop problems. These include exact methods, heuristics, and meta-heuristics. However, there has been limited studies focusing specifically on the flexible flow line.

For exact methods, branch and bound algorithms have proven to be the preferred technique for solving flow line sequencing problems to optimality. The majority of branch and bound techniques have focused on solving scheduling problems with parallel machines [10–20] and hybrid flow shops [21–25]. In 1993, Sawik [26] first proposed a mathematical model for solving small-sized scheduling problems in a flexible flow line with limited buffers. Jungwattanakita et al. [27] proposed a mixed integer programming model for minimizing the makespan and the number of tardy jobs for a flexible flow line. In Jungwattanakita et al. [27] model, they considered job sequence and machine-dependent set up times.

The NP-hardness of the flexible flow line problem underscores the need and motivation to consider heuristic and meta-heuristic approaches [28]. Brah and Loo [29] evaluated the effectiveness of various heuristic methods that had not been applied to flow shops with multiple processors. There is limited literature that applies heuristic and meta-heuristic algorithms to flexible flow line problems. The proposed random keys genetic algorithm of Kurz and Askin [30] was applied to a flexible flow line model. They minimized makespan for a system with sequence dependent setup times. Akrami et al. [31] proposed GA and Tabu search meta-heuristic methods for solving a specific flexible flow line sequencing model with limited intermediate buffers and compared results with an optimal enumeration method (OEM) on randomly generated problems. In Akrami et al. [31] flexible flow line model, sequencing, lot size, and product specific scheduling decisions are considered. They minimize total costs made up of the sum of setup costs, work-in-process inventory holding costs, and finished products inventory holding costs. Computational results from generated problems illustrated the promising performance of GA and Tabu search in comparison with OEM on large-size problems. Jenabi et al.[32] proposed a Hybrid Genetic Algorithm (HGA) and Simulated Annealing (SA) to solve flexible flow line problems with unrelated parallel processors on finite planning horizons. The main objective is to minimize the total setup and inventory holding costs per unit time without any stock-out. Final results proved the outperformance of HGA on generated problems compared with SA without considering the computation times. Quadt and Kuhn [33] proposed a nested genetic algorithm to minimize setup costs and the mean flow time in batch scheduling flexible flow lines. The nested GA consists of two parts. First, an outer GA determines a target number of setups from which batch sizes are derived. Secondly, the inner GA determines a schedule for all production stages based on the outer GA parameters. The proposed GA focused exclusively on batch scheduling in a flexible flow line. Tavakkoli-Moghaddam et al. [34] proposed a mimetic algorithm (MA) with a novel local search engine for solving a flexible flow line with process blocking. Computational results from the proposed MA on large-sized problems were found to be more effective when compared to classic GA.

In this paper, the flexible flow line with an objective of minimizing the makespan will be explained and formulated as an integer programming (IP) model. Due to the difficulty of obtaining the optimal solutions for medium and large-size problems, we use the Fuzzy Cell Genetic Algorithm for solving such problems. To the authors knowledge, no specific work relating to the general form of the flexible flow line scheduling problem using GA optimization or any other optimization method is published in literature to compare to the performance of the proposed method. Also, this benchmark is the first time a clear explanation is offered for generating problems with known solutions.

II. Flexible Flow Line Scheduling Problem Description

The mathematical formulation for the two-stage flexible flow shop with the objective of minimizing the makespan was presented by Guirchoun et.al [35]. Kurz and Askin [9] proposed the mathematical model for the flexible flow line that also considers sequence-dependent setup times.

The integer programming model formulation which addresses the problem in this paper will be shown here. Let t be the number of tasks to be schedule and k the number of parallel machines at station s. N, W and P represent sets of jobs, stages and parallel machines respectively. The problem definition is presented as:

Indices	
t	index of job to be scheduled ($t = 1, 2, \dots, n$)
s	index of stations ($s = 1, 2, \dots, m$)
k	index of machine ($k = 1, 2, \dots, p$)
mn	last station
Parameters	
$Pt_{t,s}$	processing time of job s in station t
$Ct_{s,k}$	completion time of all jobs in station s and machine k
$Ct_{mn,k}$	completion time of all jobs in the last station mn and machine k
$Ft_{t,s,k}$	finishing time of task t in machine k in station s
Decision Variables	
$x_{0,t,k}$	1 if job t performs as a first job in parallel machine k and 0 otherwise
$xn_{t,k}$	1 if job t performs as a last job in parallel machine k and 0 otherwise
$x_{i,j,k}$	1 if job i performs before job j in parallel machine k and 0 otherwise

$$\text{minimize} \quad z \tag{1}$$

$$\text{s.t.} \quad \sum_{t \in N} x_{0,t,k} = 1, \forall k \in P \tag{2}$$

$$\sum_{k \in P} x_{0,t,p} + \sum_{k \in P} \sum_{j \in N} x_{t,j,k} = 1, \quad \{\forall t, j \in N | t \neq j\} \tag{3}$$

$$x_{0,t,k} + \sum_{i \in N} x_{i,t,k} = x_{n,t,k} + \sum_{j \in N} x_{t,j,k}, \quad \forall k \in P, \{\forall i, j, t \in N | i, j \neq t\} \tag{4}$$

$$Ct_{s,k} - Ft_{t,s,k} \geq 0, \forall s \in W, \forall k \in P, \forall t \in N \quad (5)$$

$$Ft_{t,1,k} - \sum_{t \in N} Pt_{t,1} x_{0,t,k} \geq 0, \forall k \in P \quad (6)$$

$$Ft_{t,s,k} - Ft_{t,s-1,k} - \sum_{t \in N} Pt_{t,1} x_{0,t,k} \geq 0, \\ \{\forall s \in W | s > 1\}, \forall k \in P, \forall t \in N \quad (7)$$

$$Ft_{j,1,k} - Ft_{i,s,k} \geq Pt_{j,1} x_{i,j,k} - Pt_{i,1} x_{j,i,k} \\ -(M - Pt_{j,s})(1 - x_{i,j,k} - x_{j,i,k}), \{\forall i, j \in N | i \neq j\} \quad (8)$$

$$Ft_{j,s,k} - Ft_{i,s,k} \geq Pt_{j,s} x_{i,j,k} - Pt_{i,s} x_{j,i,k} \\ -(M - Pt_{j,s})(1 - x_{i,j,k} - x_{j,i,k}), \{\forall i, j \in N | i \neq j\} \quad (9)$$

$$Ft_{j,s,k} - Ft_{i,s-1,k} \geq Pt_{j,s} x_{i,j,k}, \{\forall i, j \in N | i \neq j\}, \\ \{\forall s \in W | s > 1\} \quad (10)$$

$$z \geq Ct_{mn,k}, \forall k \in P \quad (11)$$

$$x_{i,j,k} = 0, \quad \forall k \in P, \{\forall i, j \in N | i \neq j\} \quad (12)$$

$$x_{0,t,k} \in \{0, 1\}, \quad \forall k \in P, t \in N \quad (13)$$

$$x_{L,t,k} \in \{0, 1\}, \quad \forall k \in P, t \in N \quad (14)$$

$$x_{i,j,k} = 0, \quad \forall k \in P, \{\forall i, j \in N | i \neq j\} \quad (15)$$

We assume each task in each station has the same processing time in each parallel machine within a station. Also, each task must complete processing in a station before moving to the next station. The sequence of the jobs is the same from the first station to the last station. There is a restriction in this model that every stage must be visited by at least as many jobs as there are machines in that stage. Eq.(1)represents the objective function which is to minimize the makespan *z*. Eq.(2) defines the constraint to assign at least one job as a starting job to each machine in parallel for each station. Constraint (3) ensures that job t is either the first job or should be done immediately after job *j* in machine *k*. Eq.(4) represents all the eligibility of each job in each machine. Constraint set (5) ensures that the completion time of all the jobs in machine k and station s should be greater than or equal to the finishing time of each individual task in the same machine and station. In Eq.(6), finishing time of the first job assigned to machine *k* at the first station should be greater than or equal to the processing time of the first job at the same machine. Constraint set (7) enforces job t sequencing through each station after another. Eq.(8) calculates the accumulation of processing time for the set of tasks in each machine at the first station. Constraint set (9) and (10) calculate the finishing time of job *t* according to the equations (16) and (17).

If *t* is the first job in parallel *k* at station *s*:

$$Ft_{t,s,k} = Ft_{t,s-1,k} + Pt_{t,s}, \{\forall s \in W | s > 1\}, \\ \forall k \in P, \forall t \in T \quad (16)$$

Else if *t* is the successor of the task *j* in parallel *l* at station *s*:

$$Ft_{t,s,k} = \max(Ft_{t,s-1,k}, Ft_{t,s-1,k}) + Pt_{t,s}, \\ \{\forall t, j \in T | t \neq j\}, \{\forall s \in W | s > 1\}, \forall k \in P \quad (17)$$

Constraint set (11) links the decision variable z and $Ct_{mn,1}$ to minimize the completion time of the maximum machine in parallel at the last station *mn*. The last sets of equations (12),(13), (14), and (15) are defining the binary decision variables for the predecessor and successor job relations, first job, last job and zero for the case when the predecessor and successor have the same syntax in the problem.

III. Fuzzy Cell Genetic Algorithm (FCGA)

The Fuzzy Cell Genetic Algorithm (FCGA) mimics the main concepts of Meiosis and Mitosis found in human cell division. During the mitosis process, a cell with $2n$ number of chromosomes begins by growing (replicating) and the individual chromosomes start to align. Then, chromatids (one of two identical chromosomal strands) move toward the left or right side of the cell without any physical connection. Finally, the content of the cell is divided into two new daughter cells, and the chromosomes in each are replicated to two sister chromatids. The final product of mitosis is two new cells, containing $2n$ chromosomes.

In a different manner, Meiosis consists of two sub-steps in the process namely meiosis I and meiosis II. The final product of this process with four cells where each one containing two distinct chromatids. Crossover happens during Meiosis I. The final step in meiosis occurs when four cells with each one contains two distinct chromatids. Mitosis and meiosis are two distinct processes in cell division for making sexual and asexual chromosomes. All these procedures are replicated the process in human cells.

The FCGA starts with random populations (Population1 and Population2) of job sequences and the order in which they will process in each stage. Chromosomes in each population are divided into two sub-populations for the mitosis and meiosis procedures using the fuzzy logic controller. Fuzzy logic intelligently controls the number of Asexual and Sexual chromosomes contained in Subpopulation1 and Subpopulation2. Individuals sent to the meiosis sub-population are going through the crossover and mutation procedures while the chromosomes sent to mitosis are going through duplication and have a smaller probability of mutation. New chromosomes in each subpopulation will be evaluated using a fitness function which calculates the makespan corresponding to that particular job sequence.

There are three metrics to assess FCGA performance that are used for controlling and prevent it from premature convergence. $F_{average}$ defines the mean fitness distribution of the population and F_{best} (best fitness) measures the best individual (lowest fitness value for minimization problems). The worst fitness (F_{worst}) is defined as the worst individual fitness value in the population. To obtain the changing rate (F_{change}) in a population, $F_{worst} - F_{best}$ is calculated in each iteration. It should be mentioned that the fitness value is the makespan for each sequence inside the chromosome. The three metrics are used as inputs for the fuzzy controller. The controller module monitors the performance of the meiosis process. If the diversity rate inside the populations decreases, the controller will start to increase the size of Subpopulation1 for mitosis. This increases solution diversity and moves away from premature convergence. However, the increased changing rate will decrease the size of mitosis candidates in Subpopulation2 to protect form diverging away from the optimal solution. Figure 2 shows the flow chart for FCGA.

Previously, FCGA was tested on continuous benchmark functions and generated superior results when compared with other meta-heuristic methods in Shirazi et. al. [36]. Also, FCGA has shown outstanding potential for scheduling flow lines in comparison with other heuristic and meta-heuristic approaches on well-known benchmark functions [37].

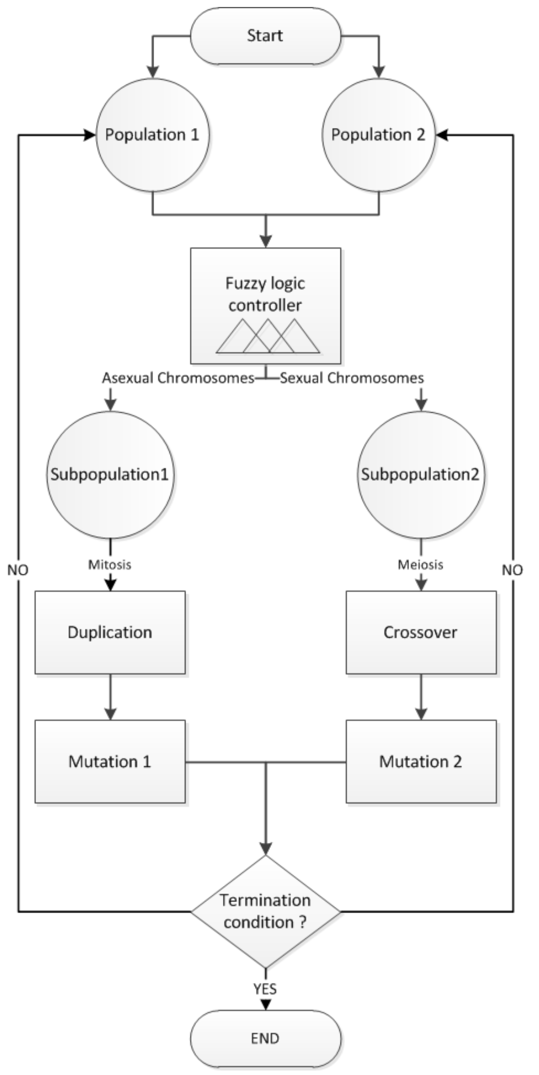

Fig. 2. flowchart: considers the proposed GA and fuzzy logic controller for adjusting the procedures in mitosis and meiosis.

IV. Generating Problem Sets

It is important to assess the performance of the FCGA on complex flexible flow line problems which IP models cannot solve in an efficient period of time. Kurz and Askin [9] proposed data generation for flexible flow shop problems while also considering setup times. However, they did not clearly explain their method for generating data sets fitting this type of flow shop.

The following section will detail how problem sets are generated. In addition, a sample of generated problem data will be examined to more closely detail how the problem generator operates.

The requirement for generating data is *n* (number of jobs), *s* (number of stages), *m*(number of machines in parallel in each stage), the range of processing time for every i_{th} job in each j_{th} stage defined by Pt_i,j and T_j,k is the sets of job in j_{th} stage and k_{th} parallel machine. It assumes the numbers of machines in each stage are the same and has to be more than one. The main purpose is to provide the sequence of jobs with no gap between the processes. The pseudocode of the problem generator is shown in TableI.

TABLE I
PSEUDOCODE OF THE PROBLEM GENERATOR FOR FLEXIBLE FLOW LINE DATA SETS

Steps	Description
1	**Initialization**
2	**Input** *n*, *s*, *m*, *L*, *U*
3	Makespan = 0
4	Generate sets with n number of jobs :$T = T_{1,1}, T_{1,2}, , T_{j,k}$
5	**For** each machine in parallel; $k = 1, 2, , l$
6	Calculate the first order job $i \in T_{j,k}$; $Pt_{1,j} \leftarrow random[L, U]$
7	$Makespan_k \leftarrow Pt_{1,j}$
8	**For** each job in stage s ; $j \in T_{j,k} \vert i > 1$
9	Calculate the process time for job *i* in machine *j*
10	$Pt_{i,j} \leftarrow random[L, U]$
11	$Makespank_k \leftarrow Makespan_k + Pt_{i,j}$
12	$Pt_{i-1,j+1} \leftarrow Pt_{i,j}$
13	**End For**
14	**End For**
15	**Set** $Makespan \leftarrow \arg max_{k \in 1,2,...,l}\{Makespan_k\}$
16	**Save** $ProcessTime \leftarrow [Pt_{i,j}]_{n\times m}$, Makespan
17	**End**

Figure 3 shows an example of 4 jobs, 2 stages and 2 machines in each stage. As it shown in the example, the final output for the problem generator is $PT_{4\times 2} = \begin{bmatrix} 4 & 6 \\ 7 & 6 \\ 6 & 4 \\ 8 & 4 \end{bmatrix}$ with a solution of 18 sec.

The problem generator provides solutions with no gap between jobs in each stage and at every machine. It should be mentioned that the generated solution is not an optimal solution. However, the solution provided is sufficient to compare with other procedures solutions. While algorithms have difficulty solving larger problems, the problem generator can easily generate solutions for any size problem.

The processing time for each job is randomly generated as an integer in the range of $[1, 10]$. There are 3 problem sets generated. The first set considers the number of $jobs(n) = 5, 8, 10, 20, 30, 40, 50, 70, 100$ and the number of $stages(s) = 2, 5, 15$ by assuming two machines in each stage. Secondly, a set of jobs (n) and a set of stages $(s) = 20, 30, 40, 50, 100$ are selected with 5 machines in each stage. Lastly, the number of $jobs(n) = 20, 40, 100$ and the number of $stages(s) =$

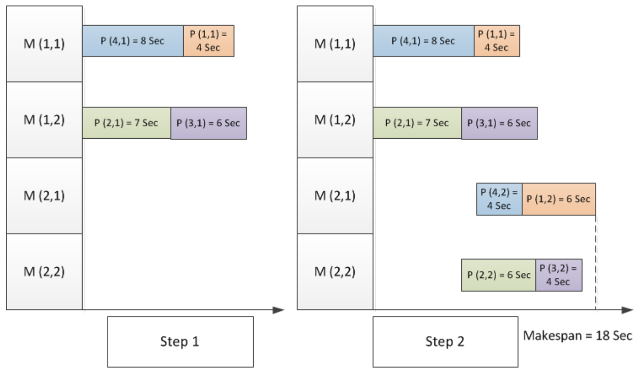

Fig. 3. An example of problem generator to provide a known solution (18sec) for 4 jobs 2 stages and 2 machines.

$10, 20, 40, 60, 100$ with 10 machines in each stage. Table II represents the parameters for the data sets.

TABLE II
PROBLEM SETS CONSIDERED

	Machine 2	**Machine 5**	**Machine 10**
#Jobs	5,8,10,20,30,40,50,70,100	20,30,40,50,100	20,40,100
#Stage	2,5,15	20,30,40,50,100	10,20,40,60,100
Processing time	Random[1,10]	Random[1,10]	Random[1,10]
#Operation ($n \times s$)	10-300	40-10000	40-10000
#Instance per set	10	10	10
#Instance total	100	50	60

V. RESULTS AND DISCUSSION

The IP models have been solved using Cplex 12 on an Intel Xeon $E5-26650$, RAM 64 GB of memory. The first stopping criterion is to reach to 0.05 for the relative optimality criteria (Optcr), where Optcr defines as:

$$\frac{|\text{Obj} - C_{best}|}{(1.0 - 10 + |C_{best}|)} < Optcr \tag{18}$$

In formulation (18) Obj is the current objective function value and C_{best} is the best proposed integer solution. The second stopping criterion is runtime, where CPU time is limited to 24 hours.

The IP model could only solve the following problem configurations $[m(jobs) \times s(stages)]$: $[5 \times 2], [8 \times 5], and[10 \times 5]$, each featuring 2 machines in parallel at each stage. FCGA also found optimal solutions for this subset of problems, excluding two variants of $[10 \times 5]$ which have been solved by GAMS[38]. Figure 4 illustrates the results of FCGA and the IP model across the aforementioned subset of problems.

The remaining 70 problems with two machines in parallel that were unable to be solved by the IP model are plotted separately in Figure 5. FCGA produced better solutions than the problem generator in 59 of the 70 problem configurations. In 2 of the other 11 instances, FCGA found the same solution

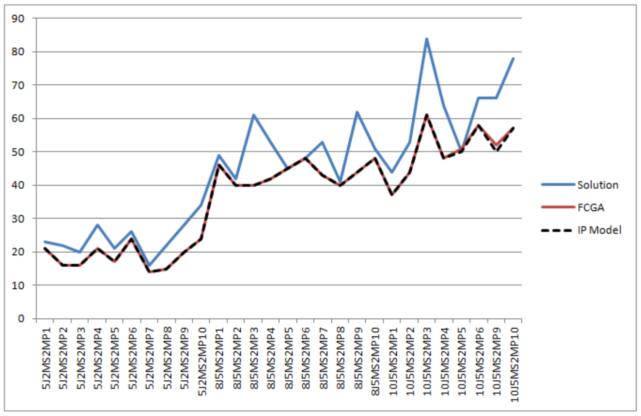

Fig. 4. Compare FCGA and problem generator solutions with IP results for measuring the algorithm performance on problems $[5 \times 2]$, $[8 \times 2]$, $[10 \times 5]$ with 2 machines in each stage.

provided by the problem generator. The FCGA has shown outstanding search performance in a solution space for small size problems. However, the problem generator hasnt shown great performance for problem sets when compared to the IP model. Therefore, the problem generator solutions do not provide a robust reference for FCGA on larger problem instances.

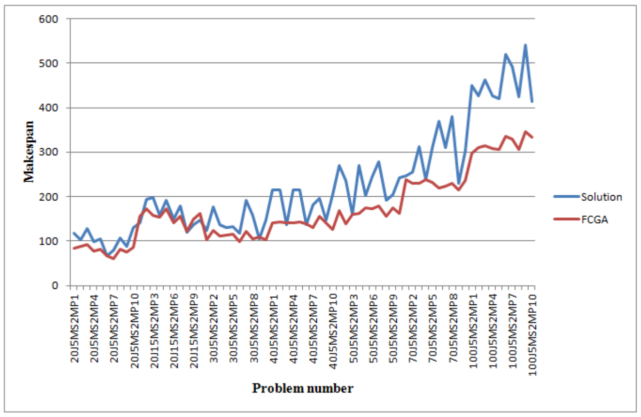

Fig. 5. Compare FCGA and problem generator solutions with IP results for measuring the algorithm performance on problems $[20 \times 5]$, $[20 \times 15]$, $[30 \times 5]$, $[40 \times 5]$, $[50 \times 5]$, $[70 \times 5]$, $[100 \times 5]$ with 2 machines in each stage.

In the second problem set featuring 5 machines in parallel, FCGA outperformed the problem generator in 16 of the 50 problem instances (Figure 6). FCGA solutions are measured relative to the best known solution via a loss function where loss is calculated as: $(makespan - C_{best})/C_{best}$. Loss is accumulated for each subset of problems to asses FCGA performance.

Accumulated loss for problem instances $[30 \times 30], [40 \times 40]$ and $[50 \times 50]$ are $0.27, 0.52,$ and 0.42 respectively. The highest total loss value for FCGA is 0.86 for $[100 \times 100]$ problem instances. On a per problem basis this represents a small amount of loss. The problem generator solutions have shown to be an efficient reference point for large size problems.

The last part is concentrated on more complex problems featuring 10 machines in each stage which is typically larger

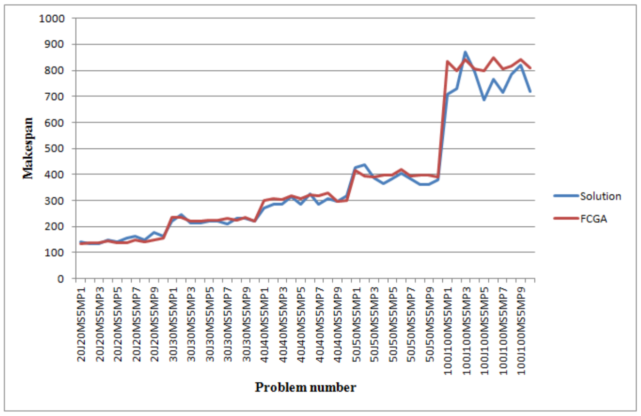

Fig. 6. Compare FCGA and problem generator solutions for measuring the algorithm performance on problems $[20 \times 20]$, $[30 \times 30]$, $[40 \times 40]$, $[50 \times 50]$, $[100 \times 100]$ with 5 machines in each stage.

than problems found in the literature. As it shown in Figure 7, FCGA performs much better for the problems with the size of $[20 \times 20]$, $[40 \times 40]$, $[100 \times 10]$,$[100 \times 20]$ and the highest total loss is less than 0.2% for all problems. However, the loss value increased by 0.77 for the problem set with size of $[100 \times 60]$. For really large size problems $[100 \times 100]$ the loss goes up to 0.77 which is still small value for the FCGA performance on that complexity.

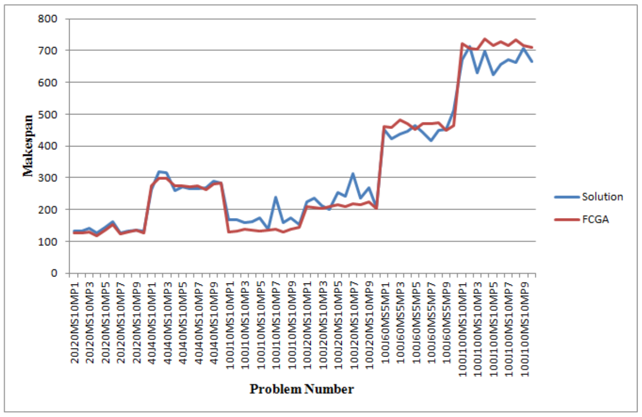

Fig. 7. Compare FCGA and problem generator solutions for measuring the algorithm performance on problems $[20 \times 20]$, $[40 \times 40]$, $[100 \times 10]$, $[100 \times 20]$, $[100 \times 60]$, $[100 \times 100]$ with 10 machines in each stage.

The problem generator could provide the superior solutions for the truly large size problem and provide robust references to assess the FCGA performance.

VI. Conclusion

The flexible flow line problem is an important representation of many real world applications. Because of the complexity of this problem, not only it is difficult to find optimal solutions, but benchmark problems are also sparse. In this research a formulation and a FCGA procedure for solving this problem has been presented and discussed. A process for generating problem set is also developed. The performance of FCGA is tested, and the results show that FCGA can be used for solving real world size problem with satisfactory solutions.

References

[1] J. Błażewicz, K. H. Ecker, G. Schmidt, and J. Weglarz, *Scheduling in computer and manufacturing systems*. Springer Science & Business Media, 2012.

[2] R. Tavakkoli-Moghaddam, N. Safaei, and F. Sassani, "A memetic algorithm for the flexible flow line scheduling problem with processor blocking," *Computers & Operations Research*, vol. 36, no. 2, pp. 402–414, 2009.

[3] A. Agnetis, A. Pacifici, F. Rossi, M. Lucertini, S. Nicoletti, F. Nicolo, G. Oriolo, D. Pacciarelli, and E. Pesaro, "Scheduling of flexible flow lines in an automobile assembly plant," *European Journal of Operational Research*, vol. 97, no. 2, pp. 348–362, 1997.

[4] S. Piramuthu, N. Raman, and M. J. Shaw, "Learning-based scheduling in a flexible manufacturing flow line," *IEEE Transactions on Engineering Management*, vol. 41, no. 2, pp. 172–182, 1994.

[5] E.-H. Aghezzaf and H. Van Landeghem, "An integrated model for inventory and production planning in a two-stage hybrid production system," *International Journal of Production Research*, vol. 40, no. 17, pp. 4323–4339, 2002.

[6] J. Grabowski and J. Pempera, "Sequencing of jobs in some production system," *European Journal of Operational Research*, vol. 125, no. 3, pp. 535–550, 2000.

[7] Z. Jin, K. Ohno, T. Ito, and S. Elmaghraby, "Scheduling hybrid flowshops in printed circuit board assembly lines," *Production and Operations Management*, vol. 11, no. 2, pp. 216–230, 2002.

[8] H. D. SHERALI, S. C. SARIN, and M. S. KODIALAM, "Models and algorithms for a two-stage production process," *Production Planning & Control*, vol. 1, no. 1, pp. 27–39, 1990.

[9] M. E. Kurz and R. G. Askin, "Comparing scheduling rules for flexible flow lines," *International Journal of Production Economics*, vol. 85, no. 3, pp. 371–388, 2003.

[10] J. Schutten and R. Leussink, "Parallel machine scheduling with release dates, due dates and family setup times," *International journal of production economics*, vol. 46, pp. 119–125, 1996.

[11] T. Edwin Cheng and M. Y. Kovalyov, "Parallel machine batching and scheduling with deadlines," *Journal of Scheduling*, vol. 3, no. 2, pp. 109–123, 2000.

[12] S. Webster and M. Azizoglu, "Dynamic programming algorithms for scheduling parallel machines with family setup times," *Computers & Operations Research*, vol. 28, no. 2, pp. 127–137, 2001.

[13] J. Blazewicz and M. Y. Kovalyov, "The complexity of two group scheduling problems," *Journal of Scheduling*, vol. 5, no. 6, pp. 477–485, 2002.

[14] M. Azizoglu and S. Webster, "Scheduling parallel machines to minimize weighted flowtime with family set-up times," *International Journal of Production Research*, vol. 41, no. 6, pp. 1199–1215, 2003.

[15] Z.-L. Chen and W. B. Powell, "Exact algorithms for scheduling multiple families of jobs on parallel machines," *Naval Research Logistics (NRL)*, vol. 50, no. 7,

pp. 823–840, 2003.

[16] B. Lin and A. Jeng, “Parallel-machine batch scheduling to minimize the maximum lateness and the number of tardy jobs,” *International Journal of Production Economics*, vol. 91, no. 2, pp. 121–134, 2004.

[17] S. Dunstall and A. Wirth, “A comparison of branch-and-bound algorithms for a family scheduling problem with identical parallel machines,” *European Journal of Operational Research*, vol. 167, no. 2, pp. 283–296, 2005.

[18] R. Nessah, C. Chu, and F. Yalaoui, “An exact method for problem,” *Computers & Operations Research*, vol. 34, no. 9, pp. 2840–2848, 2007.

[19] S.-O. Shim and Y.-D. Kim, “A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property,” *Computers & Operations Research*, vol. 35, no. 3, pp. 863–875, 2008.

[20] E. Mehdizadeh, R. Tavakkoli-Moghaddam, and M. Yazdani, “A vibration damping optimization algorithm for a parallel machines scheduling problem with sequence-independent family setup times,” *Applied Mathematical Modelling*, vol. 39, no. 22, pp. 6845–6859, 2015.

[21] S. A. Brah and J. L. Hunsucker, “Branch and bound algorithm for the flow shop with multiple processors,” *European journal of operational research*, vol. 51, no. 1, pp. 88–99, 1991.

[22] C. Rajendran and D. Chaudhuri, “An efficient heuristic approach to the scheduling of jobs in a flowshop,” *European Journal of Operational Research*, vol. 61, no. 3, pp. 318–325, 1992.

[23] O. Moursli and Y. Pochet, “A branch-and-bound algorithm for the hybrid flowshop,” *International Journal of Production Economics*, vol. 64, no. 1, pp. 113–125, 2000.

[24] D. Santos, J. Hunsucker, and D. Deal, “Global lower bounds for flow shops with multiple processors,” *European Journal of Operational Research*, vol. 80, no. 1, pp. 112–120, 1995.

[25] P. Fattahi, S. M. H. Hosseini, F. Jolai, and R. Tavakkoli-Moghaddam, “A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations,” *Applied Mathematical Modelling*, vol. 38, no. 1, pp. 119–134, 2014.

[26] T. J. Sawik, “A scheduling algorithm for flexible flow lines with limited intermediate buffers,” *Applied stochastic models and data analysis*, vol. 9, no. 2, pp. 127–138, 1993.

[27] J. Jungwattanakit, M. Reodecha, P. Chaovalitwongse, and F. Werner, “Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria,” *The International Journal of Advanced Manufacturing Technology*, vol. 37, no. 3-4, pp. 354–370, 2008.

[28] J. N. Gupta, “Two-stage, hybrid flowshop scheduling problem,” *Journal of the Operational Research Society*, vol. 39, no. 4, pp. 359–364, 1988.

[29] S. A. Brah and L. L. Loo, “Heuristics for scheduling in a flow shop with multiple processors,” *European Journal of Operational Research*, vol. 113, no. 1, pp. 113–122, 1999.

[30] M. E. Kurz and R. G. Askin, “Scheduling flexible flow lines with sequence-dependent setup times,” *European Journal of Operational Research*, vol. 159, no. 1, pp. 66–82, 2004.

[31] B. Akrami, B. Karimi, and S. M. Hosseini, “Two meta-heuristic methods for the common cycle economic lot sizing and scheduling in flexible flow shops with limited intermediate buffers: The finite horizon case,” *Applied Mathematics and Computation*, vol. 183, no. 1, pp. 634–645, 2006.

[32] M. Jenabi, S. F. Ghomi, S. A. Torabi, and B. Karimi, “Two hybrid meta-heuristics for the finite horizon elsp in flexible flow lines with unrelated parallel machines,” *Applied Mathematics and Computation*, vol. 186, no. 1, pp. 230–245, 2007.

[33] D. Quadt and H. Kuhn, “Batch scheduling of jobs with identical process times on flexible flow lines,” *International Journal of Production Economics*, vol. 105, no. 2, pp. 385–401, 2007.

[34] R. Tavakkoli-Moghaddam, N. Safaei, and F. Sassani, “A memetic algorithm for the flexible flow line scheduling problem with processor blocking,” *Computers & Operations Research*, vol. 36, no. 2, pp. 402–414, 2009.

[35] S. Guirchoun, P. Martineau, and J.-C. Billaut, “Total completion time minimization in a computer system with a server and two parallel processors,” *Computers & Operations Research*, vol. 32, no. 3, pp. 599–611, 2005.

[36] A. N. Shirazi, S. Manbir, and H. Alashwal, “Modified genetic algorithm based on human cell mechanisms,” *Soft Computing*, in prep.

[37] A. N. Shirazi, M. Steinhaus, and M. Sodhi, “Scheduling flow line by fuzzy cell genetic algorithm,” *International Journal of Production Research*, in prep.

[38] A. K. Brooke and A. D Meeraus, “Gams release 2.25; a user’s guide,” GAMS Development Corporation, Washington, DC (EUA), Tech. Rep., 1996.