PREDICTIVE-REACTIVE JOB SHOP SCHEDULING FOR FLEXIBLE PRODUCTION SYSTEMS WITH THE COMBINATION OF OPTIMIZATION AND SIMULATION BASED ALGORITHM

J.Y. Tan[1], A.A. Abdul Rahman[1], M.A.A. Rahman[1], M.R. Salleh[1] and P. Bilge[2]

[1]Faculty of Manufacturing Engineering,
Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya,
76100 Durian Tunggal, Melaka, Malaysia.

[2]Departments of Machine Tools and Factory Management,
Berlin Institute of Technology,
Straße des 17. Juni 135, 10623 Berlin, Germany.

Corresponding Author's Email: [1]azrulazwan@utem.edu.my

Article History: Received 15 September 2020; Revised 2 November 2020; Accepted 16 December 2020

ABSTRACT: A significant issue for the production sector was the complicated scheduling requirement due to shorter product life cycles and unexpected fluctuations. Scheduling has a significant effect on the ability of a manufacturing system to meet the deadlines and the schedule should be reactive to resolve disturbances during operation. Yet, job shop scheduling issues are nondeterministic polynomial time - hard (NP-hard). This research addresses some aspects of combining simulation and optimization-based algorithms for job-shop scheduling and rescheduling of flexible production systems. The predictive part determines the feasible schedule to be used for a flow shop which is generated using a combination of rule-based simulation and optimization: first, using the optimization algorithm to compute a rough plan, followed by using a rule based simulation system to locally fine tune the plan to obtain the final schedule. The schedule obtained will be implemented to the real-world system which is adapted by the reactive part of the system. The results had proved that the predictive-reactive scheduling can effectively increase the effectiveness of flexible production system. It would be a promising approach to combine the advantages of simulation with optimization algorithm.

KEYWORDS: *Simulation; Optimization; Genetic Algorithm; Predictive-Reactive; Job-Shop Scheduling*

1.0 INTRODUCTION

The rising number of products and variants, together with random input orders and non-standardized manufacturing processes, forcing the existing production systems to become more complicated [1-3]. The manufacturing time for a product has increased due to the limitations and sophistication of the processing processes [4]. Production systems need to handle market volatility, adopt to new products, and order changes rapidly [5-6]. Flexible production systems have recently become very popular in research, and most algorithms, dispatching rules and strategies have already been developed [7-8]. However, traditional methodical methods are complex in formulation; furthermore, due to the sophistication and a great number of variables and restrictions, most available algorithms do not give a reliable outcome [9–13].

Therefore, the schedule must be predictive-reactive, and combination of simulation and optimization algorithm can deal with uncertainties and disruptions during the reactive phase. First objective will be fixed to determine the feasible schedule which is the predictive part of the systems; second objective will be to create an algorithm with combination of simulation and optimization using a created model based on the feasible schedule and flexible production system's layout; and third objective will be to validate the algorithm with the reactive-part of the systems.

Hence, the focus of this research is to build the simulation model for flexible production cells which can provide versatility in system layout and product mix with flexible routing and production sequence.

2.0 METHODOLOGY

2.1 Input Data

The following data from Tables 1 to 5 are inserted as input for simulation process. Table 1 shows that the reference process plan for simulation runs. There are total 10 type of products (Type_1 to Type_10) for production, and each of the product required several different processes to complete. There are total 5 type of processes (Proc_1, Proc_2, Proc_3, Proc_4 and Proc_5) while for product Type_6, Type_7 and Type_8 is flexible whether to or not to maintain the process sequences during the production process.

Table 1: Process plan

Name	Keep Sequence	1st Process	2nd Process	3rd Process	4th Process	5th Process
Type_1	True	Proc_1	Proc_3	Proc_4		
Type_2	True	Proc_2	Proc_5	Proc_1	Proc_4	
Type_3	True	Proc_5	Proc_2	Proc_4	Proc_3	
Type_4	True	Proc_3	Proc_4	Proc_5	Proc_1	Proc_2
Type_5	True	Proc_2	Proc_4	Proc_3	Proc_1	
Type_6	False	Proc_2	Proc_3	Proc_4		
Type_7	False	Proc_1	Proc_4	Proc_5		
Type_8	False	Proc_2	Proc_4			
Type_9	True	Proc_5	Proc_3	Proc_1		
Type_10	True	Proc_2	Proc_3	Proc_5		

Table 2 shows data for the processing time in seconds for each type of products based on the type of process. The empty column indicate that the specific process is not available for the specific products, for example, process 1 (Proc_1) is not required for production of product Type_3, Type_6, Type_8 and Type_10. The last row of Table 2 indicates the process is carried out at which workstation, for example, process 1 (Proc_1) will only carried out in workstation H1 and H2.

Table 2: Workplaces or stations process and processing time (minutes)

Product name \ Process	Proc_1	Proc_2	Proc_3	Proc_4	Proc_5
Type_1	1200		1800	420	
Type_2	1500	900		600	240
Type_3		660	510	735	420
Type_4	180	120	300	135	210
Type_5	1830	1200	915	495	
Type_6		600	720	1200	
Type_7	600			900	600
Type_8		840		1500	
Type_9	1080		480		1200
Type_10		2100	660		600
Workstation	H1, H2	H2	H3	H4	H3, H4

Table 3 shows the number of each product to be process and the due dates. The due date is written in format of date, then followed by the time. All the products were required to finish before the due date with the specific quantity, for example, Type_1 product is required to produce for 5 quantities before first of May 2020, 1.10 pm.

Table 3: Order of the products

Product Name	Qty	Due Date
Type_1	5	01.05.20 13:10
Type_2	10	01.05.20 11:35
Type_3	12	02.05.20 10:55
Type_4	4	01.05.20 14:55
Type_5	8	02.05.20 16:15
Type_6	10	01.05.20 11:40
Type_7	3	01.05.20 9:40
Type_8	6	01.05.20 20:40
Type_9	15	02.05.20 11:40
Type_10	2	01.05.20 18:40

Table 4 shows the approximate downtime of stations, which indicate that which workstation is in use and not idle, the date and time that the workstation start the production process and the duration left that the specific process of the workstation is about to end.

Table 4: Workplaces downtime

Workplace	Start	Duration
H1	01.05.20 8:30	1:30:00.0000
H2	01.05.20 8:15	1:00:00.0000

Table 5 shows list of products that are still in the production process. The work order number is the sequence of the specific product, for example, Type_1 product is the fifth product transported into the system. The due date and the location of the product is also provided, together with the number of process had been carried out (nProc) and the process sequence of the specific product.

Table 5: In-process products

Work Order No.	Name	Due Date	Location	nProc	Process Sequence
5	Type_1	01.05.20 8:10	H3_Buffer	2	123
6	Type_2	01.05.20 8:25	H1_Buffer	3	1234
7	Type_3	01.05.20 8:55	H2_Buffer	2	1234
8	Type_4	01.05.20 8:45	H4_Buffer	2	12345
9	Type_5	01.05.20 8:20	H2_Buffer	1	1234
10	Type_2	01.05.20 8:30	H1_Buffer	3	1234

2.2 General Scheduling Procedure

The process of scheduling was simplified into a flow chart as showed in Figure 1. Stage 1 in scheduling process included the preparations of data input. Stage 2 was the process of searching a schedule that fulfilled the requirements to achieve optimum result by run several simulations

with different dispatching rules with genetic algorithm. After a suitable schedule was obtained, Stage 3 was implemented. Simulation was run to obtain logs to control a real system and to check the performance measures. However, if there were station failure or planned maintenance, the rescheduling request might be triggered. Hence, reactive loop represented the transfer of system status information to the model and reactivation of scheduling procedure with new inputs.

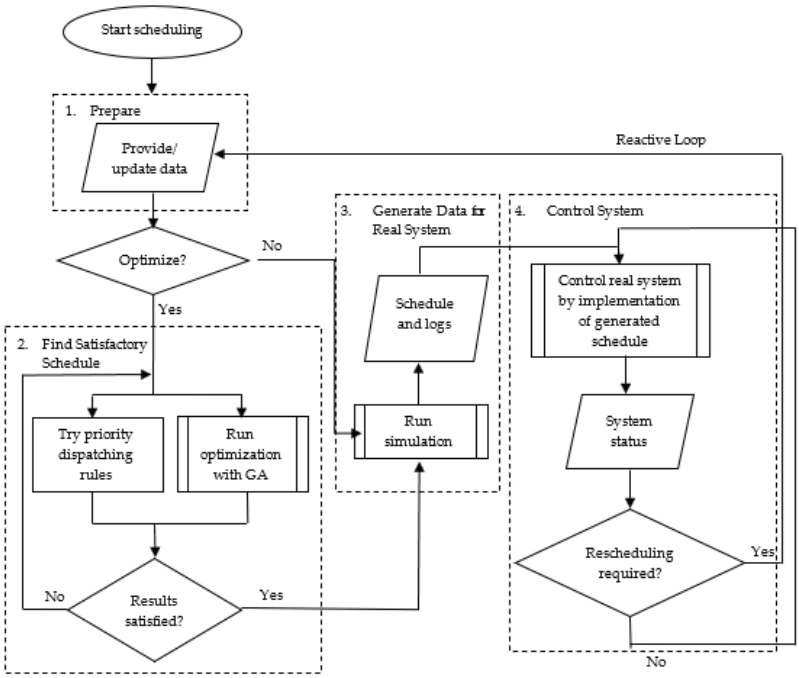

Figure 1: Scheduling procedure

2.3 Model Description

The developed model for both case studies were illustrated in Figure 2 and Figure 3. Both models were run by two sets of methods, the first set of method was used to control the model for simulation purpose, while the second set was used to prepare input data for simulation purpose.

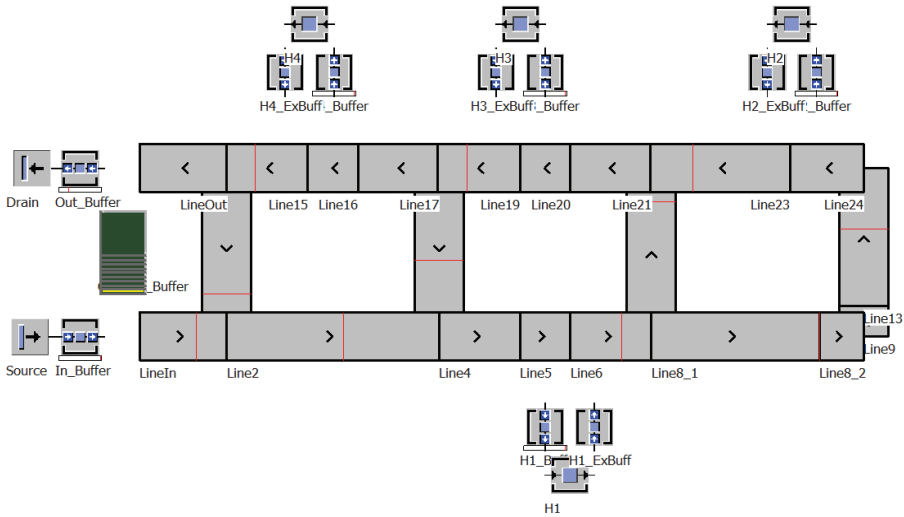

Figure 2: Simulation model layout with conveyor line and workstations

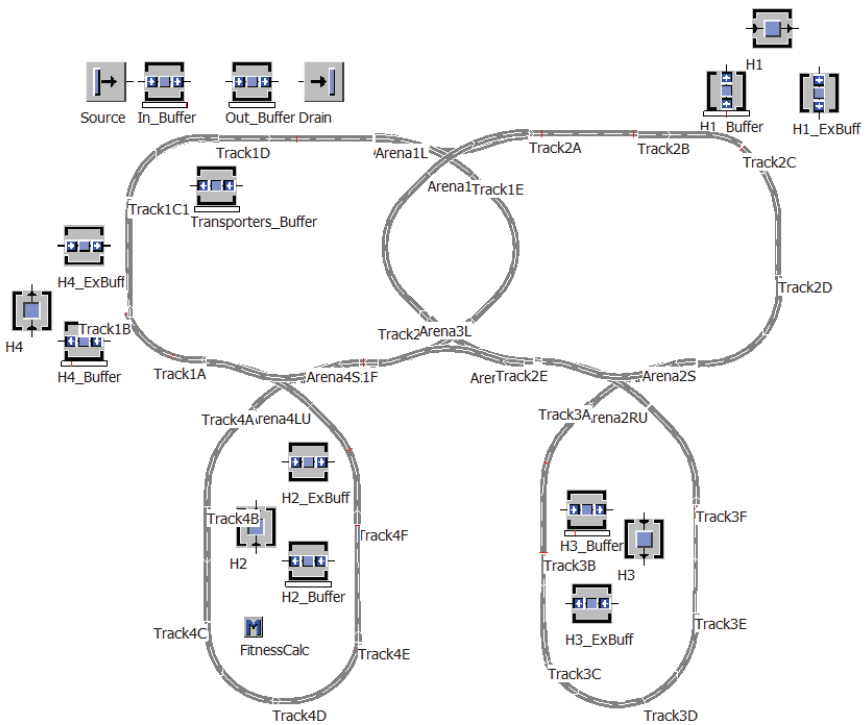

Figure 3: Simulation model layout with tracks and workstations

2.3.1 Control Methods

The main methods that used to control the objects during simulation process were as below:

i. Move, which is used to perform transportation of carriers and transporters for pick-up and return tour at workstations and buffers by following the path table.
ii. LineStopper, which is used to stop the line when the next line was occupied and decide which carrier is going to move first according to priority.
iii. Load and Unload, which is used to load and unload carriers and transporters at the entrance of the system together with the entities released by source and the end of the system.
iv. Call_Carrier or Call_Transporter, which is used to call the carriers and transporters when an entity started to process at workstation, based on the calculated transported time and processing time needed.

2.3.2 Model Initialization Methods

The initialization methods were methods that worked as transforming initial input data into data that used for simulation process, such as:

i. Path_Generator, which is used to generated predefined paths for empty carriers and transporters based on the system layout.
ii. Create_Prod_List, which is used to transform initial input data into tables that were needed for model controlling.
iii. FindShortestPath, which is used to choose the shortest path among all the possible paths generated.

2.4 Generation of Paths

Genetic algorithm (GA) was chosen to be used for the optimization process in the model, however Tecnomatix Plant Simulation Version 12 has a ready-to-use wizard that directly applied genetic algorithm concept when the wizard is used. Permutation was the method that used to get all possible sequences of the processes and triggered when the product was not necessary to follow the process plan. This method was programmed to receive a string of characters, for example 1234, and returned a list of permuted strings. Figure 4 shows the terminology process of permutation to be performed. Furthermore, complexity levels of paths were set to filter out the longer paths and narrow down the field of possibilities for genetic algorithm mechanism to determine

better schedule when optimization time is limited. The higher level indicated the larger number of possible paths to be generated.

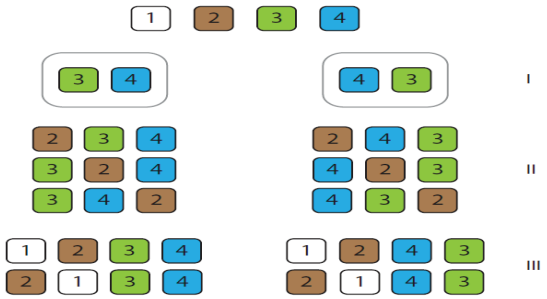

Figure 4: Permutation algorithm

3.0 RESULTS AND DISCUSSION

3.1 Multi-Process Performance

The optimization process was run under four different streams with two types of generations to identify the effect of number of streams towards the time consumed for optimization process. The existed research had been proved that the smaller number of generations produced the best results [14]; while multiple streams in GA can reach the optimal solution faster than a single run [15].

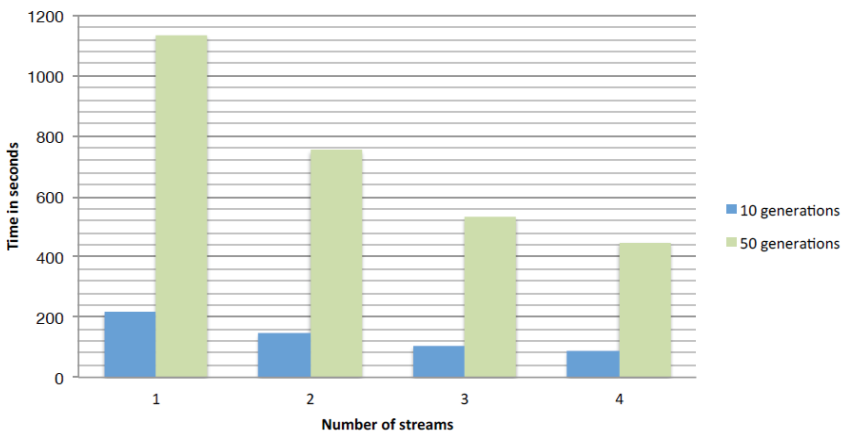

Figure 5: Multi-process performance

The results showed that a smaller number of generations consumed less time, while the time consumed in both cases of generations run with the number of streams of four resulted 2.5 times lower than in single stream. The graph in Figure 5 clearly indicated that the decline

pattern of the time consumed during optimization process with the increased number of streams, which is consistence with the existing research. This can be concluded that the performance was increased in multiple streams compared to in single stream as the time consumed is getting lesser. The findings and results were validated with experiments on functions of varying difficulty [15].

3.2 Release and Control Option Comparison

The model was run with combinations of different options for pickup (CB11, CB01, CB00) and source control option (Opt0, Opt1, Opt2, Opt3) to analyze which options was the most optimized. An existed research claimed that the fitness value generated by selection process in genetic algorithm indicated how good the solutions as shown in [16].

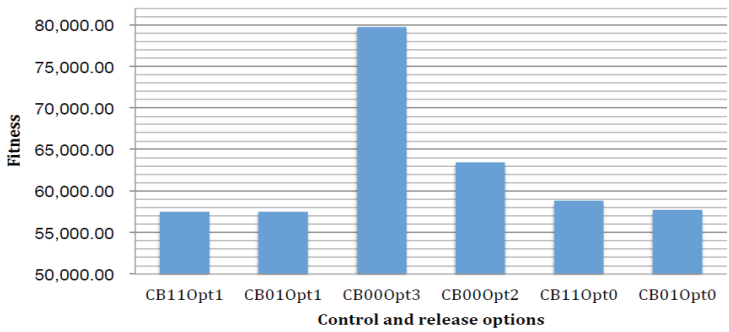

Figure 6: Best schedules comparison

The result showed in Figure 6 indicated that the first combination (CB11Opt1), second combination (CB01Opt1), fifth combination (CB11Opt0) and the sixth combination (CB01Opt0) showed similar performance with similar fitness value, while the fourth combination (CB00Opt2) showed slightly better performance with slightly higher fitness value. The third combination (CB00Opt3) resulted in the highest fitness value among the others. Hence, this can be concluded that the third combination which had the highest fitness value provide the best material flow. The roulette wheel selection had been used as benchmark for validations of the fitness value [17].

3.3 Improvement Rate

Vrajitoru [18] had proved that the smaller number of generations size bring to the most optimal result faster. The optimization was run multiple time with the number of generations fixed to 20 but with vary generation size. The rate was calculated based on the following equation:

$$\text{Rate} = \frac{\text{Initial Fitness} - \text{Optimized}}{\text{Time Consumed}} \quad (1)$$

The result showed in Figure 7 proved that the pattern of the improvement speed is decreasing with increase of generation size, which indicated that the larger the generation size, the lower the improvement rate. This result is corresponding with the findings from Vrajitoru [18]. However, there is an increase in improvement rate of generation size of 15, 20 and 45 compared to their previous result. Although there is an increase improvement rate but still the smallest generation size showed the largest improvement rate. This result can be concluded that the smaller generation sizes showed better performance while longer runs showed the opposite. Experiment approach and theoretical analysis were used to validate the most optimum generation size as shown in [18].

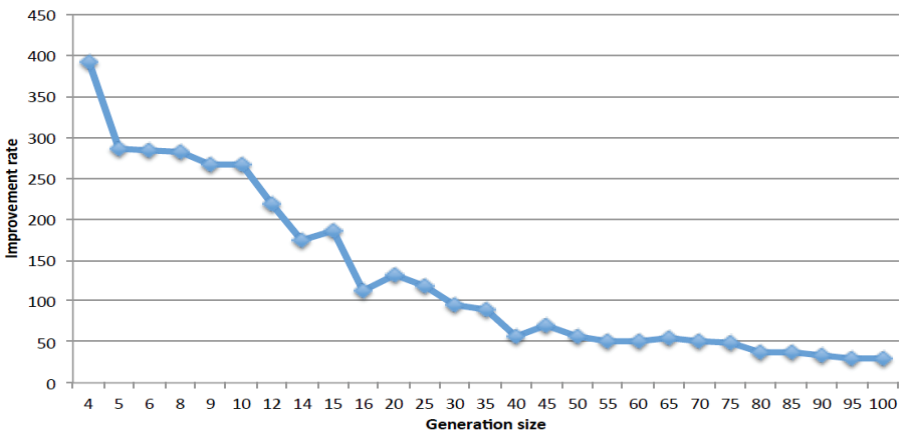

Figure 7: Improvement rate in 27 generations

3.4 Paths Complexity Levels and Layout Complexity

The model was tested with an increased complexity layout to test whether the algorithm can fit with a more complex layout system. The model was added more lines and two more workstations. After running the simulation, the result shown that the model could work as well as the original layout even with more options and variants.

The increased complexity layout was used for analysis for the performance of paths complexity levels. The optimization was run for 10 minutes but different path complexity level filters. The result shown in Figure 8 indicated that the complexity level 2 showed slightly better

performance as it had achieved 3% better fitness value compared to level 1 in the longer run of 50 generations. However, level 3 was slightly outperformed than level 2. From this result, complexity level 2 was the best option compared to others, however, conclusion can be made that the lowest complexity level is better choices compared to lowest paths complexity level, because of increasing number of paths choices.

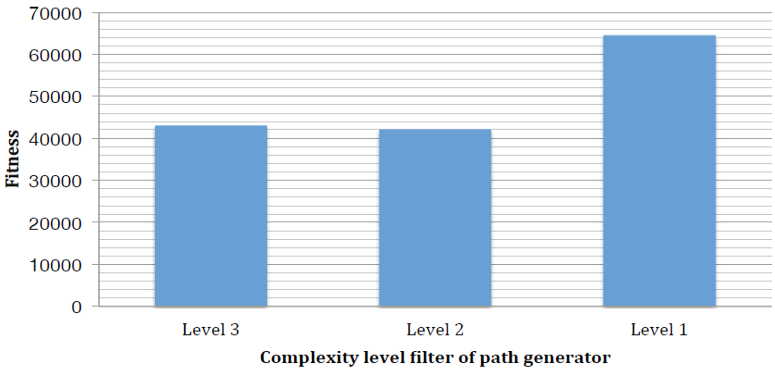

Figure 8: Performance of complexity level filters

4.0 CONCLUSION

This research aimed to analyze several aspects of combining simulation and optimization-based algorithms for job-shop scheduling of flexible production systems. For predictive part, the feasible schedule will be determined for further analysis from a developed model. The result proved that the model established demonstrated good efficiency and the ability to find an effective schedule in a specified period. The combination of simulation and optimization algorithm can lead to produce the best result in a more effective way. The results had proved that the complex scheduling problem can be solved effectively and consumed lesser time with the algorithm. Lastly, the schedule will be validated by the real-world system which is the reactive part. The results shown that even with increase complexity layout and paths, the schedule can perform optimization and obtain the best result effectively. Overall, model demonstrated the success of combining simulation and optimization with genetic algorithm and given the desired flexibility and control capability to engineer.

ACKNOWLEDGMENTS

Authors are grateful to Universiti Teknikal Malaysia Melaka for the financial support through FRGS/1/2017/TK03/FKP-SMC/F00342.

REFERENCES

[1] A. Allahverdi, E. Pesch, M. Pinedo and F. Werner, "Scheduling in manufacturing systems: new trends and perspectives", *International Journal of Production Research*, vol. 56, no. 19, pp. 6333–6335, 2018.

[2] L. Asadzadeh, "A local search genetic algorithm for the job shop scheduling problem with intelligent agents", *Computers & Industrial Engineering*, vol. 85, pp. 376–383, 2015.

[3] M. Niehues, F. Buschle and G. Reinhart, "Adaptive job-shop control based on permanent order sequencing", *Procedia CIRP*, vol. 33, pp. 127–132, 2015.

[4] B. Scholz-Reiter, D. Lappe and S. Grundstein, "Capacity adjustment based on reconfigurable machine tools – harmonising throughput time in job-shop manufacturing", *CIRP Annals*, vol. 64, no. 1, pp. 403–406, 2015.

[5] R. Angkiriwang, I. N. Pujawan and B. Santosa, "Managing uncertainty through supply chain flexibility: reactive vs. proactive approaches", *Production & Manufacturing Research*, vol. 2, no. 1, pp. 50–70, 2014.

[6] Z. Ebrahim and A. A. Rasib, "Unnecessary overtime as a component of time loss measures in assembly process", *Journal of Advanced Manufacturing Technology*, vol. 11, no. 1, pp. 37-48, 2017.

[7] D. Y. Lee and F. DiCesare, "Integrated scheduling of flexible manufacturing systems employing automated guided vehicles", *IEEE Transactions on Industrial Electronics*, vol. 41, no. 6, pp. 602–610, 1994.

[8] N. Bhatt and N. R. Chauhan, "Genetic algorithm applications on job shop scheduling problem: a review", in International Conference on Soft Computing Techniques and Implementations, United States, 2016, pp. 7–14.

[9] X. Q. Wan and H. S. Yan, "Integrated scheduling and self-reconfiguration for assembly job shop in knowledgeable manufacturing", *International Journal of Production Research*, vol. 53, no. 6, pp. 1746–1760, 2015.

[10] Y. C. Choi and P. Xirouchakis, "A holistic production planning approach in a reconfigurable manufacturing system with energy consumption and environmental effects", *International Journal of Computer Integrated Manufacturing*, vol. 28, no. 4, pp. 379–394, 2015.

[11] M. Niehues, P. Sellmaier, T. Steinhaeusser and G. Reinhart, "Adaptive job-shop control using resource accounts", *Procedia CIRP*, vol. 57, pp. 351–356, 2016.

[12] M. M. Nasiri, R. Yazdanparast and F. Jolai, "A simulation optimisation approach for real-time scheduling in an open shop environment using a composite dispatching rule", *International Journal of Computer Integrated Manufacturing*, vol. 30, no. 12, pp. 1239–1252, 2017.

[13] T. Nehzati, N. Ismail, F. A. Aziz and S. A. Hosseini, "Research outline on reconfigurable manufacturing system production scheduling employing fuzzy logic", *International Journal of Electronics*, vol. 2, no. 5, pp. 813-816, 2012.

[14] M. S. Gibbs, H. R. Maier, G. C. Dandy and J. B. Nixon, "Minimum number of generations required for convergence of genetic algorithms", in IEEE International Conference on Evolutionary Computation, Vancouver, Canada, 2006, pp. 565–572.

[15] E. Cantú-Paz and D. E. Goldberg, "Are multiple runs of genetic algorithms better than one?", in 5th Annual Genetic and Evolutionary Computation Conference, Chicago, United States, 2003, pp. 801–812.

[16] P. Bajpai and M. Kumar, "Genetic algorithm–an approach to solve global optimization problems", *Indian Journal of Computer Science and Engineering*, vol. 1, no. 3, pp. 199–206, 2010.

[17] R. Yadav and W. Ahmad, "Benchmark function optimization using genetic algorithm", *Journal of Engineering, Computers & Applied Sciences*, vol. 2, no. 6, pp. 21–24, 2013.

[18] D. Vrajitoru, "Large population or many generations for genetic algorithms? Implications in information retrieval", in *Soft Computing in Information Retrieval*, F. Crestani and G. Pasi. Heidelberg: Physica-Verlag, 2000, pp. 199–222.