Algorithm for Solving Job Shop Scheduling Problem Based on machine availability constraint

Kanate Ploydanai
Department of Industrial Engineering, Faculty of Engineering, Kasetsart University, Bangkok, Thailand

Anan Mungwattana
Department of Industrial Engineering, Faculty of Engineering, Kasetsart University, Bangkok, Thailand

***Abstract*—Typically, general job shop scheduling problems assume that working times of machines are equal, for instance eight hours a day. However, in real factories, these working times are different because the machines may have different processing speeds, or they may require maintenance. That is, one machine may need to be operated only half day whereas other machines may have to be operated for the entire day. So, each machine has its own working time window. In this paper, this type of problem is referred to as a job shop scheduling problem based on machine availability constraint which is more complex than typical job shop scheduling problems. In the previous research, this type of problem has been rarely investigated before. Thus a new algorithm is developed based on a non-delay scheduling heuristic by adding machine availability constraint to solve job shop scheduling problem with minimize makespan objective. The newly developed algorithm with the machine availability constraint assumption is more realistic. The study reveals the result of algorithm that consider machine availability constraint is better than the result of algorithm that ignores machine availability constraint when apply to the real problem.**

Keywords-Job shop scheduling; algorithm; heuristic; optimization; non-delay scheduling; machine availability constraint

I. INTRODUCTION

Developing algorithms for solving job shop scheduling problems is a popular research in the field of optimization. It has been known that the job shop scheduling problems are NP hard [1]. That means when the size of problem grows up, the time for determining the optimal solutions of the problem increase exponentially.

Typically, there are M machines and N jobs for scheduling. Jobs have to be processed on these machines with different routes or sequences. So, the complexity of scheduling depends on number of machines, number of jobs, and sequences of jobs. There are $N!$ ways (solutions) to sequence jobs on each machine. For some type of scheduling problem, all machines use the similar sequence. So, the numbers of feasible solutions to find the optimal solution are $N!$ solutions. For the job shop scheduling problem, each machine has different sequences; therefore, number of feasible solution increase to $(N!)^M$ solutions.

Many researches developed algorithm for solve job shop scheduling problem. In-Chan Choi [2] aimed to develop local search algorithm to solve job shop scheduling problem. The objective function is to minimize makespan. Sequence dependent setup condition is added to the problem. The setup time of each job depends on sequence of jobs in each machine. This paper solves the problem by local search algorithm. Local search algorithm helps to reduce computation time. D.A. Koonce [3] used data mining to find the pattern of schedule for job shop scheduling problem. Propose of this work is to apply data mining methodology to explore the pattern. The objective of the problem is minimizing makespan. Genetic algorithm is used to generate the good solution. The data mining is used to find relationship of sequence and predict next job in sequence. The result from data mining can use to summarize new dispatching rule that gives the result likes result of genetic algorithm. Chandrasekharan [4] presented three new dispatching rules for dynamic flow shop problem and job shop problem. The performance of the rules present by comparison with 13 dispatching rules. The case study is from simulation study for flow shop scheduling problem. The problems are modified again by random routing of jobs. The problems are changed the flow shop scheduling problem to job shop scheduling problem. The study could be concluded that the performance of dispatching rules is being influenced by routing of jobs and shop floor configurations. Hiroshi [5] used shift bottleneck procedure to solve job shop scheduling problem. The objective of problem is to minimize total holding cost. The specific constraint is added to the problem. The added constraint is no tardy job constraint. The experiment show that shift bottleneck procedure can reduce computation time. Anthony [6] presented Memmetic algorithm for job shop with time lag. The time lag means minimal and maximal between start times of operations. This article presented framework to solve job shop scheduling problem base on disjunctive graph to modify the problem and solve by Memmetic algorithm. Jansen [7] solved job shop scheduling problem under assumption that jobs have controllable processing time. That means he can reduce processing time of job by paying certain cost. Jansen presented two models. The first is continuous model and the other is decrease model. The evident of proofing could be showed that

both of them can solve by Approximation scheme in polynomial time when number of machines and number of operations are fixed. Job shop scheduling problem with the minimizing makespan is investigated. Guinet [8] reduced the problem from job shop to flow shop problems by using precedence constraint of jobs. After that extended of Johnson's rule is define to solve this problem. Also, he noted that The optimality of extended Johnson's rule is proved for two machine and the rule efficient for some three of four machine job shop problems. Drobouchevitch [9] presented two heuristic to solve a special case of job shop scheduling. The case bases on assumption that each job consists of at most two operations. One of which is to be processed on one of m machines. Whie the other operation must be perform on a single bottleneck machine. One of both heuristics guarantees a worst case performance ratio 3/2. In addition, he noted that those techniques can be applied to the related problem, such as flow shop scheduling problem with parallel machines. Ganesen [10] solved the special case of job shop scheduling problem. Minimum competition time variance (CTV) constraint is added to the problem. The lower bound of CTV is developed for the problem. For solving this problem backward scheduling approach is used. To show performance of the backward scheduling approach, the result is compared with forward scheduling approach. The study showed backward scheduling approach is well performance for this special case of job shop scheduling problem. In addition two layers technique is a technique to solve the job shop scheduling problem. Pan [11] described binary mixed integer programming for the reentrant job shop scheduling problem and solves the problem by using two layers technique Ganesen [12] studied job shop scheduling problem with two objectives. The first is to minimize total absolute difference of completion time and the other is mean flow time. The backward scheduling technique was studied again. Moreover, he used static optimum technique. In this study, 82 problems were taken to study. The result is a new benchmark for the problem. Pham [13] solved special case of job shop scheduling problem namely multi mode blocking job shop scheduling problem. The problem is from hospital in order to allocate hospital resource for surgical case. CPLEX was employed to solve the problem. Because of computation time limit, the model is capable for small and medium size of problem. That is the study suggested.

Other researches investigated meta- heuristic. Watanabe [14] and Koonce [15] used genetic algorithm to solve job shop scheduling problem. Ganesen used Simulated annealing to solve job shop scheduling [16], [17]. Some research used neural network to select dispatching rule. Some researches solve job shop scheduling by hybrid algorithm between two Meta heuristics [19], [20], [21].

This paper focuses on developing algorithm to solve job shop scheduling problem. The algorithm is designed by considering machine availability constraint. Next section describes detail of problem and mathematic modeling of problem. Next, machine availability constraint is described. The machine availability constraint is used to calculate realistic makespan for factories that have breaking period during processing time.

II. PROBLEM STATEMENT AND MATHEMATICAL FORMULATION

In the previous section, we reviewed the algorithms used for solving the job shop scheduling problems. This section presents the general job shop scheduling mathematical model and the detail of machine availability constraint. In general, variable are as follows:

Let $t_{i,j}$ be start time of job j that is perform on machine i,

Let $f_{i,j}$ be finish time of job j that is performed on machine i,

Let $p_{i,j}$ be processing time of job j that is performed on machine i,

Let $C_{\max}$ be Makespan (finish time of latest job).

The objective of the problem is to minimize makespan. The mathematical model of job shop scheduling problem without machine availability constraint is shown below.

$$\text{Min } C_{\max} \tag{1}$$

St

$$t_{h,j} - t_{i,j} \geq p_{i,j} \tag{2}$$

$$C_{\max} - t_{i,j} \geq p_{i,j} \tag{3}$$

$$t_{i,j} - t_{i,k}. \geq p_{i,k} \text{ or } t_{i,k} - t_{i,j} \geq p_{i,j} \tag{4}$$

$$t_{i,j} \geq 0 \tag{5}$$

$$t_{i,j} \geq r_i \tag{6}$$

$$t_{i,j} + p_{i,j} \leq d_j \tag{7}$$

To make sure that the next step on machine h of job j starts after finish time of the step on machine i of job j, equation 2 is employed. Next, equation 3 ensures that C_{Max} must be more than finish time of the last job. Equation 4 is used for sequencing jobs on the machines. This equation means that only one job can be processed only one machine at a time. By using equation 5, the start time of processes is non negative. Some time, some problem requires condition of job released time. Equation 6 ensures that a job must start after the released time. The last constraint is used to control that jobs must be finished before their due dates.

The mathematical model is the general for job shop scheduling problem, but the job shop scheduling problem based on machine availability constraint is more complicated. In

addition, we develop procedure to calculate finish time of process and makespan.

Let

α be index of working day type.

β be index of breaking period

$\mu_{\alpha,\beta}$ be time to stop machine of working day type α at period β.

$\gamma_{\alpha,\beta}$ be time to start machine)again(of working day type α at period β.

Factory has many type of working day. The paper suggests to keep information of working time by using $\mu_{\alpha,\beta}$ and $\gamma_{\alpha,\beta}$ by α as index of working day type and β as index of breaking period in the working day type. For example, working day type 1 has two breaking period. First breaking period is lunch period ($\mu_{1,1} = 12:00$ pm and $\gamma_{1,1} = 1:00$ pm). The second breaking period is between 4:00 pm to 8:00 am of tomorrow. That means $\mu_{1,2}$ equals 4:00 PM and $\gamma_{1,2}$ equals 8:00 AM. For case of second breaking period $\mu_{\alpha,\beta}$ more than $\gamma_{\alpha,\beta}$ that mean $\mu_{1,2}$ is the time of next day. On the other hand, if $\mu_{\alpha,\beta}$ less than $\gamma_{\alpha,\beta}$, $\mu_{\alpha,\beta}$ and $\gamma_{\alpha,\beta}$ are on the same day.

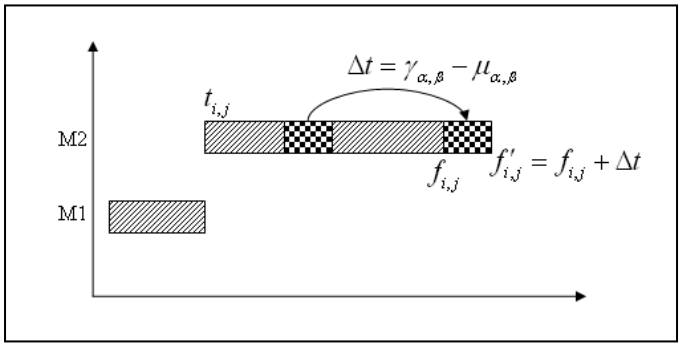

Figure 1. Finish time computation with a breaking period.

When breaking period occurs in processing time, the calculation of finish time is different form the typical calculation. Figure 1 illustrates how to compute finish time when breaking period is in processing time. Working period start from $t_{i,j}$ and finish at $f_{i,j}$. If a breaking period is in the working time, algorithm compensates breaking time by $\Delta t = \gamma_{\alpha,\beta} - \mu_{\alpha,\beta}$ and finish time is calculated by $f'_{i,j} = f_{i,j} + \Delta t$. Also, if there are two working period times are in working time, the finish time is calculated as Figure 2. There are many cases of breaking period between processing periods of job operations. Breaking period appears many types between processing time.

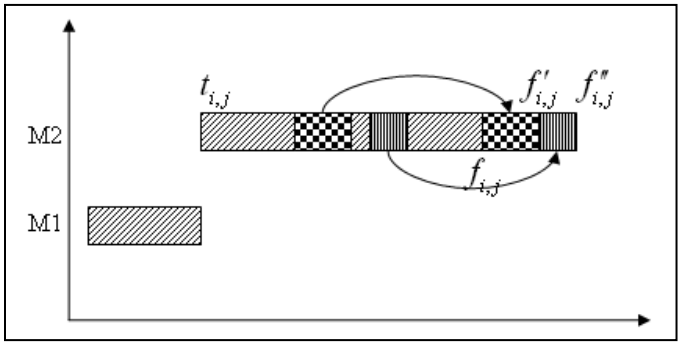

Figure 2. Finish time computation of two breaking period.

Indeed, this section presents the mathematic model of general job shop scheduling, and presents machine availability constraint in a factory briefly. Job shop scheduling with machine availability constraint is a special problem. In the next section, we present non delay-scheduling algorithm. For special case of problem, we present an algorithm to calculate finish time with machine availability constraint. The algorithm to calculate finish time of job by considering machine availability constraint names working time window algorithm.

III. ALGORITHMS FOR JOB SHOP SCHEDULING PROBLEM

In this section, we present a non-delay scheduling algorithm (NSA), working time window algorithm (STA), and the algorithm that are combined between the non delay scheduling algorithm and working time window algorithm. We call the new algorithm that working time window non-delay scheduling algorithm (WT-NSA).

First, NSA is presented. NSA uses the concept that a machine is never idle when its queue is not empty [22]. Computational time required for this algorithm can be minimal. For *M* machines and *N* jobs, it spends *MN* computational time to solve the problem. The solution by this algorithm, however, is not the optimal solution. Nonetheless, the problem is NP-Hard. The optimal solution might not be possible to determine in a short time.

A. Non-delay scheduling algorithm.

The procedure of NSA is to compute all operational times of all jobs that are able to process in the list. Next, the operation with the earliest finish is chosen. Then, the operations that are able to be processed are updated to the list and the same procedure will continue again until all operations are selected. The procedure of NSA [22] can be shown as followed:

At state t ,let S_t be the partial schedule of $(t-1)$ operations. Let A_t be the set of operations schedulable at stage t, that is, all predecessor operations are in S_t .Let e_k be

the earliest time that operations $k \in A_t$ can be scheduled, that is, predecessor are completed and machine is available.

Non-delay schedule generation.
Step:

Step 0 Initialize. Let $t = 1, S_1 = \phi$. A_1 contains the first operation of each ready job

Step 1 Select operation. Find $e^* = \min_{k \in A_t}$
If several e^* exist, choose arbitrarily. Let m^* be machine need by e^* .Choose any $k \in A_t$ that requires m^* and has $e_k = e^*$ (If all non delay schedules are to be created, choose all such k and create a new partial schedule for each)

Step 2 Increment. Add the selected operation k to s_t to create s_{t+1} .Remove k from A_t and add the next operation for its job unless that job is completed; this creates A_{t+1} .Set $t = t + 1$.If $t = MN$ stop; otherwise go to 1

B. Working time window

In this section, we present algorithm for calculating finish time with machine availability constraint. It is well suitable for factories that have many machines, many jobs, and several breaking period. The algorithm is called working time window algorithm.

Working day includes working period and breaking period. Working period is period of time to operate jobs on machines. Breaking period is period of time to stop operating.

If breaking period does not occur between operation times, the finish time of the operation equal start time pluses processing time. Otherwise, if some part of breaking period is in operation time, the finish time is compensated.

The developed algorithm is recursive algorithm. Algorithm calls itself until the finish time is complete to calculate. For iteration, for example, algorithm checks breaking period and compensates this time. Sometimes, an operation cannot process for a day. Algorithm calls itself one time for calculating again. Working time window calculation is shown as followed.

Algorithm1: working time window algorithm ($t_{i,j}$, $f_{i,j}$, $p_{i,j}$)

$$f_{i,j} = t_{i,j} + p_{i,j}$$

SET

$$b_1 = \begin{cases} true & ; t_{i,j} \le \mu_{\alpha,\beta} \le f_{i,j} \\ false & ; otherwise \end{cases}$$

$$b_2 = \begin{cases} true & ; t_{i,j} \le \gamma_{\alpha,\beta} \le f_{i,j} \\ false & ; otherwise \end{cases}$$

$$b_3 = \begin{cases} 1 & ; \mu_{\alpha,\beta} \le t_{i,j} \le \gamma_{\alpha,\beta} \\ 0 & otherwise \end{cases}$$

$$b_4 = \begin{cases} 1 & ; \mu_{\alpha,\beta} \le f_{i,j} \le \gamma_{\alpha,\beta} \\ 0 & otherwise \end{cases}$$

/*to calculate new finish time by rules followed that:*/

If $\sim b_1 \wedge b_2 \wedge b_3 \wedge \sim b_4$ then
$t_{i,j} = \gamma_{\alpha,\beta}$ and $f_{i,j} = f_{i,j} + (f_{i,j} - \mu_{\alpha,\beta})$
Else If $b_1 \wedge b_2 \wedge b_3 \wedge \sim b_4$ then
$t_{i,j} = \gamma_{\alpha,\beta}$ and $f_{i,j} = f_{i,j} + (\gamma_{\alpha,\beta} - \mu_{\alpha,\beta})$
Else If $b_1 \wedge b_2 \wedge \sim b_3 \wedge \sim b_4$ then
$t_{i,j} = \mu_{\alpha,\beta}$ and $f_{i,j} = f_{i,j} + (\gamma_{\alpha,\beta} - \mu_{\alpha,\beta})$
Else If $b_1 \wedge \sim b_2 \wedge \sim b_3 \wedge b_4$ then
$t_{i,j} = \mu_{\alpha,\beta}$ and $f_{i,j} = f_{i,j} + (\gamma_{\alpha,\beta} - \mu_{\alpha,\beta})$
End if

If $e <$ finish time of working time window then

Finish time = $f_{i,j}$

Else

Call working time window procedure ($t_{i,j}$, $f_{i,j}$,0) again

End if

First, the finish time is approximated by the start time and processing time. Next breaking period information ($\mu_{\alpha,\beta}$ $\gamma_{\alpha,\beta}$) are considered. The algorithm sets β the first breaking period between the start time and finish time of operation. b_1, b_2, b_3 , and b_4 are set for identify state of breaking period and time to compensate. After that we use if-then structure for recalculating finish time of job operation. Last, if the finish time of the job operation is more that the end of working day, then algorithm calls itself again. The iteration occurs until the finish time of the job operation less than the end of current working day.

In addition, this paper presents working time window algorithm and presents the mixed algorithm between Non-delay scheduling algorithm and working time window

algorithm. The mixed algorithm is called WT-NSA. This mixed algorithm looks similar to the NSA, but it is different. For WT-NSA, we modify procedure to calculated finish times of job operations.

Algorithm 2 shows the detail of WT-NSA. First, algorithm calculates finish times of all operations for each job if the operation is ready to operate at this time. Next step, the algorithm check and recalculates finish time by using working time window algorithm. This step is new added to algorithm to use in the specific problem.

In addition, the operation that has earliest start time is chosen. If many jobs have the same start time, the algorithm chooses the operation with the earliest finish time. In addition, algorithm updates the operations that are ready be processed on machines. The chosen operation is recorded in the sequence for updating finish time when the operation times change. The iteration runs until all operation of jobs complete.

Algorithm 2: Non-delay scheduling algorithm based on working time window.

While all job complete
{

For J = 1 to last job

Calculate $f_{i,j}$ of current operation of job j

(by using working time window procedure)

Next j

Select job j by min $t_{i,j}$

If start time $t_{i,j}$ is equal then choose by min $f_{i,j}$

Updating operation of chosen job j

Save sequence of all operations of jobs follows for update times later

} Loop

Next section, we compare two results from experiment. The first one is result from the NSA and the other is result from the WT-NSA.

IV. EXPERIMENT

To illustrate that WT-NSA is well performed for job shop scheduling, 40 cases are used in the experiment. Almost of cases are from internet. The processing time is between zero and one hundred.

First non delay scheduling algorithm is used. After that we investigate the effect of solution when we use the solution from algorithm that not considers machine availability constraint to the problem with machine availability constraint. Three makespans of experiment are shows. The first is makespan from algorithm that ignores machine availability constraint. The second makespan form the sequence of previous but recalculate by using machine availability constraint. The second makespan use to illustrate the different time of result from the real problem in factory and the result of elder (original) algorithm. The last makespan is from the algorithm that considers machine availability constraint. The sequence of the last is different from the first and the second. The quality of solution is better because is the makespan of the last is shorter than both of them.

Table 1 shows data of the experiment. The first column shows number of problem. The second column presents number of job and third column presents number of machine. The makespan of NSA are shown in the fourth column. We fix sequence from the fourth column and recalculate by apply machine availability constraint. The new makespan are shown in the fifth column.

Next, we present WT-NSA in sixth column. New technique brings to the new sequence and new quality of solution in sixth column. The seventh column is different value from fourth and fifth columns. The eighth column is the improved value of makespan. The improved value is value that fifth column minuses by sixth column.

From the table, the average error between NSA and NSA-WT is 2676.2 for 15 jobs 15 machines problem. The average error is 3501.8 for 20 jobs 15 machines problem and is 3662.8 for 20 jobs 20 machines problem. Last, the average error is 5169.5 for 30 jobs 15 machines problem. Almost data shows that if the problems are applied by algorithm that ignores machine availability constraint, the result has too much error.

TABLE I. COMPARISON OF EXPERIMENTAL DATA

problem	job	machine	NSA	NSA-applied	WT-NSA	error	improve	% improve
1	15	15	1462	4105	3751	2643	354	9.437483
2	15	15	1446	3701	3395	2255	306	9.013255
3	15	15	1495	3926	2930	2431	996	33.99317
4	15	15	1708	4188	2877	2480	1311	45.5683
5	15	15	1618	4449	3252	2831	1197	36.80812
6	15	15	1522	4526	2917	3004	1609	55.15941
7	15	15	1434	3442	3037	2008	405	13.33553
8	15	15	1457	3831	2751	2374	1080	39.25845
9	15	15	1622	5362	3440	3740	1922	55.87209
10	15	15	1697	4693	3260	2996	1433	43.95706
11	20	15	1865	5243	3558	3378	1685	47.35807
12	20	15	1667	5037	3667	3370	1370	37.36024
13	20	15	1802	5809	4189	4007	1620	38.67271
14	20	15	1635	4918	3813	3283	1105	28.97981
15	20	15	1835	5339	3735	3504	1604	42.94511
16	20	15	1965	5520	4068	3555	1452	35.69322
17	20	15	2059	5963	3927	3904	2036	51.84619
18	20	15	1808	4919	3877	3111	1042	26.87645
19	20	15	1789	5643	4137	3854	1506	36.40319
20	20	15	1710	4762	3682	3052	1080	29.33188
21	20	20	2175	6067	4656	3892	1411	30.30498
22	20	20	1965	6118	4867	4153	1251	25.70372
23	20	20	1933	5674	4091	3741	1583	38.6947
24	20	20	2230	5719	4469	3489	1250	27.97046
25	20	20	1950	5724	4431	3774	1293	29.18077
26	20	20	2188	6003	4263	3815	1740	40.81633
27	20	20	2096	5664	3907	3568	1757	44.97057
28	20	20	1968	5344	4737	3376	607	12.81402
29	20	20	2166	5396	4407	3230	989	22.44157
30	20	20	1999	5589	4101	3590	1488	36.28383
31	30	15	2335	7318	5909	4983	1409	23.84498
32	30	15	2432	7595	4835	5163	2760	57.08376
33	30	15	2453	7834	4915	5381	2919	59.38962
34	30	15	2434	7884	5234	5450	2650	50.63049
35	30	15	2497	8150	5385	5653	2765	51.34633
36	30	15	2445	7934	5416	5489	2518	46.49188
37	30	15	2664	8219	5482	5555	2737	49.92703
38	30	15	2155	7100	4888	4945	2212	45.25368
39	30	15	2477	7479	5349	5002	2130	39.82053
40	30	15	2301	6375	5222	4074	1153	22.07966
							AVG	36.82297

Moreover, we can present that the result is improved obviously when the problems are applied by new algorithm that designed for machine availability constraint. The average of percentage improvement is calculated by averaging the different value of the fifth column and sixth column. For 15 jobs 15 machines problem, the average of percentage improvement is 34.24 percentages. For 20 jobs 15 machines problem, the average of percentage improvement is 37.54 percentages. For 20 jobs 20 machines problem, the average of percentage improvement is 30.92 percentages. For 30 jobs 15 machines problem, the average of percentage improvement is 44.59 percentages.

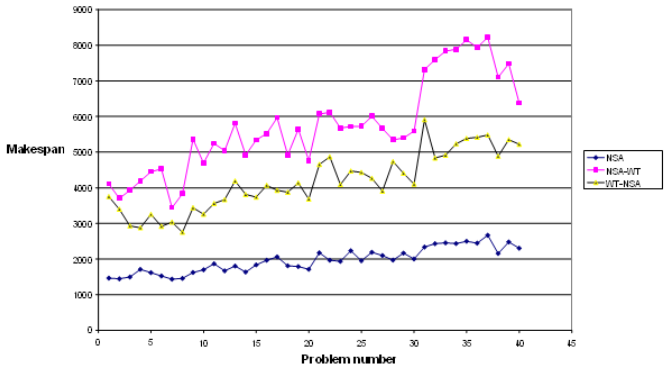

Figure 3. Graph of makesapn from three algorithms

Figure 3 depicts trend of makespan for problems. The horizontal axis presents number of problem. The first column from table 1 is the value in the horizontal axis. The vertical axis presents makespan of each problem. The makespan in forth column is used to draw graph NSA. The makespan in fifth column is used to draw graph NSA-WT. The makespan in sixth column is used to draw graph WT-NSA.

From this graph, graph from NSA is good quality value because the makespan is from algorithm that ignores machine availability constraint. Graph NSA-WT is bad quality value. The graph is from the sequence of NSA that applied machine availability constraint. Graph WT-NSA is from the algorithm that design for machine availability constraint. The result is better form graph NSA-WT.

The lowest graph is from data in forth column. The middle graph is from data in sixth column. The highest graph is from fifth column. Definitely, the WT-NSA is well performance when compare with NSA. The experiment suggests that we could design algorithm by considering machine availability constraint when the machines in the factory perform depend on machine availability constraint.

V. CONCLUSION

In this paper, we propose that algorithm could be designed by considering machine availability constraint. Data and graph form the experiment illustrate algorithm that consider machine availability constraint give the good result for all of test problems. So, the factories could use algorithm that designed by consider machine availability constraint if machines work and stop depending on period of working time. In addition, we suggest that other algorithm that use to factories that have machine availability constraint could be designed by considering machine availability constraint. In future research, the machine availability constraint is extended to the flexible job shop scheduling problem. The future problem is high complexity. We add machine availability constraint condition to the problem and solve by new developed algorithm.

REFERENCES

[1] F. S. Al-Anzi,Y. N. Sotskov, A. Allahverdi, and G. V. Andreev, “Using mixed graph coloring to minimize total completion time In job shop scheduling problem”, Applied mathematics and computation vol.182, pp. 1137-1148, 2006.

[2] I. Choi and D. Choi, “A local search algorithm for job shop scheduling problems with alternative operations and sequence dependent setups”, Computer and industrial engineering, vol. 42, pp. 43-58, 2002.

[3] D. A. Koonce and S. C. Tsai, “Using data mining to find patterns in genetic algorithm to a job shop schedule”, Computer and industrial engineering vol. 38, pp. 361-374, 2000.

[4] O. Holthaus and C. Rajendran, “A comparative of study of dispatching rules in dynamic flow shops and job shops”, European journal of operational research vol.116, pp. 156-170, 1990.

[5] H. Ohta and T. Nakatani, “A heuristic job shop scheduling algorithm to minimize total holding cost of completed and in progress products subject to no tardy job”, International journal production economics , vol. 11, pp. 19-29, 2006.

[6] A. Caumond, P. Lacomme, and N. Tchernev, “A memmetic algorithm for job shop with time lag”, Computer and operational research.

[7] K. Jansen, M. Mastrolilli, and R. S. Oba, “Approximation scheme for job shop scheduling problem with controllable processing time”, European journal of operational research, vol. 1672, pp. 97-319, 2005.

[8] A. Guinet, “Efficiency of reduction of job shop to flow shop problems”, European journal of operational research, vol. 125 pp. 469-485, 2000.

[9] I. G. Drobouchevitch and V. A. Strusevich, “Heuristic for the two stage job shop scheduling problem with a bottleneck machine”, European journal of operational research, vol. 123, pp. 229-240, 2000.

[10] V. Kumar Ganesen, A. I. Sivakumar, and G. Srinivasan, “Hierarchical minimization of completion time variance and makespan in jobshops”, Computer and operations research, vol. 33, pp. 1345-1367, 2006.

[11] C. H. Jasan and J. S. Chen, “Mixed binary integer programming formulations for the reentrant job shop scheduling problem”, Computer and operational research, vol. 32, pp. 1197-1212, 2005.

[12] V. K. Ganesen and A. I. Sivakumar, “Scheduling in static jobshops for minimizing mean flowtime subject to minimum total deviation of job completion times”, International journal production economics, vol. 103, pp. 633-647, 2006.

[13] D.N. Pham and A. Klingert,” Surgical case scheduling as a generalized job shop scheduling problem ”, Europian journal of operational research, vol.185, pp 1011-1025, 2008.

[14] M. Watanabe, K. Ida, and M. Gen, “A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem”, Computer and industrial engineering, vol. 48, pp. 743-752, 2005.

[15] D.A. Koonce and S.-C. Tsai, “Using data mining to find patterns in genetic algorithm solutions to a job shop schedule”, Computer and industrial engineering, vol. 38, pp. 361-374, 2000.

[16] V. K. Ganesen, A. I. Sivakumar, “Hierarchical minimization of completion time variance and makespan in jobshops, Computer and operational research, vol. 33, pp.1345-1367, 2006.

[17] V. K. Ganesen and A. I. Sivakumar, “Scheduling in static jobshops for minimizing mean flowtime subject to minimum total deviation of job

completion times", International journal production economics, vol. 103, pp. 633-647, 2006.

[18] A. El-Bouri and P. Shah, "A neural network for dispatching rule selection in a job shop", International journal advance manufacturing technology, vol. 31, pp. 342-349, 2006.

[19] P. D. D. Dominic, S. Kaliyamoorthy, and R. Murugan, "A conflict-based priority dispatching rule and operation-based approaches to job shops", International journal advance manufacturing technology, vol. 24, pp. 76-80, 2004.

[20] H. Ohta and T. Nakatani, "A heuristic job-shop scheduling algorithm to minimize the total holding cost of completed and in-process products subject to no tardy jobs", International journal production economics, vol. 101, pp. 19-29. 2006.

[21] W. Xia and Z. Wu, "A hybrid particle swarm optimization approach for the job-shop scheduling problem", International journal advance manufacturing technology, vol. 29, pp. 360-366, 2006.

[22] Askin and Standridge, Modeling and analysis of manufacturing systems, John Wiley and sons Inc. 1993.

AUTHORS PROFILE

Kanate Ploydanai currently is Ph.D student at the department of Industrial Engineering at Kasetsart University and he is lecturer at Industrial Engineer technology at College of industrial technology as King Mongkut 's University of Technology North Bangkok. His research interest includes production planning and control, and operation research.

Dr. Anan Mungwattana currently is an associate professor at the department of Industrial Engineering at Kasetsart University. He is also the chairman of the department. He earned the Ph.D. degree in Industrial and Systems Engineering from Virginia Tech. His research interest includes lean manufacturing, logistics and supply chain management, and facility design.