015-0855

Cost Effective Handling of Disruptions in Production Management

Ehsanallah Naseri Onur Kuzgunkaya

Department of Mechanical & Industrial Engineering

Concordia University

Montreal, H3G 1M8, Canada

Email: {e_naseri, onurk}@encs.concordia.ca

(514)-848-2424

POMS 21st Annual Conference

Vancouver, Canada

May 7 to May 10, 2010

Abstract

Rescheduling is an essential operating task to efficiently tackle uncertainties and unexpected events frequently encountered in flexible manufacturing systems. The main purpose of this paper is to develop a scheduling methodology in order to create new cost-effective schedules capable of quickly responding to such occurring disturbances and uncertainties. Responding to each disturbance by rescheduling all the remaining jobs can cause system nervousness and increased operational costs. Towards this end, a Filtered-Beam-Search-heuristic algorithm (FBS) is proposed to generate a pre-specified number of suboptimal schedules. Thereafter, a cost-based analysis method is suggested to determine the cost of changing the existing schedule to each of the alternative schedules generated by the FBS algorithm against the cost of not responding to a disturbance. Afterward, the existing schedule is replaced by the alternative schedule that generates the minimum cost impact. For this purpose, a compound cost function which consists of three cost metrics, i.e., job-related, machine-related, and material-related rescheduling costs, is developed. The results show that the rescheduling cost function can be used as a surrogate measure to minimize the actual cost of schedule changes.

Keywords

Flexible manufacturing system, Rescheduling, Filtered-beam-search, Rescheduling cost.

1. Introduction

Today's competitive manufacturing environment forces companies to be responsive to changes in the market and satisfy the need for mass customization through flexibility and adaptability in order to survive and be globally successful. Flexible manufacturing systems possess the capability of handling these disturbances thanks to the machine and process planning flexibilities; however, these capabilities should be efficiently exploited through scheduling rules in order to get the full benefit at minimal operational costs. In a dynamic environment, the task of managing and controlling manufacturing systems becomes more difficult as a result of internal disturbances such as machine failures, and external disturbances such as rush orders and supplier problems. The omission of this dynamic nature and stochastic events in scheduling literature create a gap between scheduling theory and practice. Once an initial schedule is disrupted, it should be updated through rescheduling activities to satisfy the new requirements.

Rescheduling refers to finding a new schedule upon the occurrence of a disruption in the operations of an on-going initial schedule. During the execution of a schedule, there are two important factors regarding to rescheduling as a result of disturbance occurrence. First, the timing for a rescheduling decision needs to be made. Regarding the timing decision, previous studies implemented a periodic, event-driven, or a hybrid rescheduling method which means that the schedules are updated periodically and/or whenever significant disruptions take place [1]. Sabuncuoglu and Karabuk [2] showed that frequently updating the schedule performs actually worse than myopic dispatching rules. Once the decision on updating the schedule is made, the second decision is how to perform the rescheduling action [1]. Existing studies use one of the three main rescheduling methods according to a performance criterion: (i) full rescheduling, (ii) partial rescheduling, and (iii) right-shift rescheduling. Abumaizar and Svestka [3] have developed a repair algorithm where only the operations affected by a machine breakdown are rescheduled with an objective of minimizing the deviation between the initial schedule and the updated one. One of the reasons for repairing the schedules as opposed to full rescheduling is to decrease the computation time.

Another important factor in rescheduling decisions is the performance criterion to generate the updated schedule. As a performance measure, the majority of rescheduling models optimize a time related objective function, such as tardiness, and makespan; and stability based metrics which minimize the deviation from the initial schedule [4, 5]. The main reason for the popularity of the makespan or tardiness functions is due to the fact that their primary objective is to satisfy customer needs. However, this approach may create substantial change from the initial planned schedule and may result in operational complexities in the shop floor such as reallocating tools, expediting material orders, re-setting equipment. Stability based metrics try to minimize these changes by adhering to the original schedule. However, this also might create additional operational costs such as idle machines in case of an order cancellation. Updating the schedule by only considering the time based performance measure may result in creating feasible yet impractical schedules due to high rescheduling costs. Therefore, a rescheduling cost based performance metric can be useful in assessing the negative impact of schedule updates while meeting the due dates. In this paper we propose a cost based performance criteria that will evaluate the updated schedule based on the resulting rescheduling cost.

In the case of an increased number of disturbances during the execution of a schedule, responding to every disturbance by *full rescheduling* method may create high operational cost and cause the high degree of nervousness to the system. Therefore, in some cases modifying an existing schedule may create less rescheduling cost in comparison and helps the system to be more stable. Thus, in this study, an adaptive rescheduling method is proposed to check the potential cost of switching between *do nothing* and *full rescheduling* at the time of each disturbance and the existing schedule is replaced with the one that generates the least potential cost of switching.

2. Problem Description

In this study we consider a flexible manufacturing system with partial flexibility. There are a certain number of jobs to be scheduled each having a different number of operations with alternative machines capable of performing the same operation albeit with different processing times. The initial schedule is generated by a makespan based objective function [6]. During the execution of the schedule, the following types of disturbances are considered: machine breakdown, job cancellation, and new order arrival. Two types of method of responding to disturbances considered in this study: *full rescheduling* and *do nothing*. *Full rescheduling* means rescheduling all the remaining operations which have not begun yet and *do nothing* refers to right shifting or left shifting the remaining operations depending on the disturbance type. New schedules will be generated based on the availability of the machines and remaining available jobs while minimizing the rescheduling costs. The initial schedule and updated ones by *full rescheduling* method are performed using filtered beam search method (FBS) in which the job sequence and machine allocation are determined simultaneously according to the objective function. The *do nothing* method has a different procedure depending on the disturbance type and will be explained in the following section. The following section will describe the FBS based methodology and the proposed rescheduling cost function to generate updated schedule.

3. Methodology

3.1 Filtered beam search

Filtered Beam Search (FBS) is an extension of Beam search which is the adaptation algorithm of branch and bound used in solving optimization problems. This algorithm uses heuristics to estimate certain number of best paths and eliminate permanently the rest. This algorithm works much faster than B&B as the large parts of search tree are pruned accumulatively. The beam search is like a breadth-first algorithm as it progresses level by level without backtracking [2]. However unlike breadth-first search, it doesn't search through all possible nodes and only moves down form the best promising nodes at each level. An evaluation function used to identify the promising nodes in each level, which introduces the problem of finding proper trade-off between quick but poor, and computationally demanding but better solutions [10]. Filtered beam search is introduced to find a good tradeoff between speed and accuracy [11]. By two phase evaluations which are called as local and global evaluation, filtering phase and beam selection (known as rough and accurate), nodes are pruned in each level and best node is identified. Two key parameters in FBS algorithm are *filterwidth* and *beamwidth* which identify the number of filtering nodes and the number of final solutions respectively. In first phase, keeping *filterwidth* numbers of node is done through local evaluation, and in next phase by running a global evaluation on remaining nodes the best promising node for every *beamwidth* is selected. The selected nodes in each level added to partial schedules and finally *beamwidth* number of schedules will be generated.

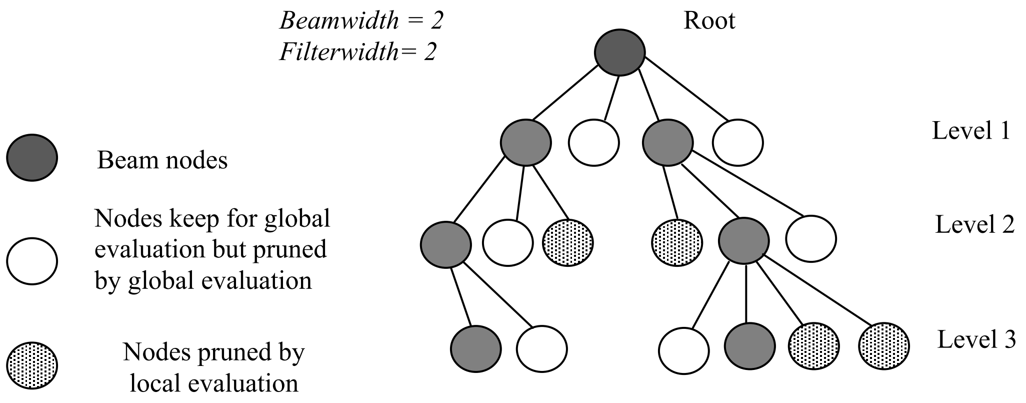
Beamwidth = 2
Filterwidth= 2
Root
Level 1
Level 2
Level 3
Beam nodes
Nodes keep for global evaluation but pruned by global evaluation
Nodes pruned by local evaluation

Fig1. Filter beam search tree representation

As shown in Figure1, after determining beam nodes in level one by global evaluation, the filtered beam search is employed independently to generate a partial schedule from each of them (in this example, as beamwidth is set to 2, two independent tree is generated would result in two schedules). Once the best node in each level for each beam is identified, nodes are generated for the next level by applying branching method. The generated nodes first locally evaluated and filterwidth numbers of nodes remain for global evaluation. The procedure continues until all the machine job pairs are allocated and beamwidth numbers of schedules are generated.

3.2 Proposed methodology

The procedure of generating schedules by FBS method consists of two phases: Generating the search space called branching methodology, and evaluating the nodes by using the global and local evaluation functions. In order to generate a search tree in FBS, two procedures called *active* and *nondelay* for job shop scheduling are discussed [7]. In this study the modified form of Nondelay called M_Nondelay [2], is used as a branching method. After a level is formed by M_Nondelay algorithm, it is ready to be bounded and is followed by search method. The key point in utilizing FBS is choosing the proper evaluation functions. Searching among available nodes is performed by these functions and partial schedules are generated at each level of the search tree. In this study, the Shortest Processing Time (SPT) is employed as a dispatching rule to generate full schedules quickly in order to assess each node with respect to local and global evaluation functions.

Finding an appropriate value for *filterwidth* and *beamwidth* is a tradeoff between accuracy and speed. The number of beamwidth determines the number of schedules to be generated for each problem, whereas the filterwidth determines the number of operation-machine allocation choices at each level. Their values are problem specific and can be determined by analyzing the tradeoff between the performance level of the generated schedule and the computation time.

In case of a disturbance at time t_0, the system will decide to do *full rescheduling* action or *do nothing*. In order to generate the updated schedule, the remaining operations and availability of each machine need to be determined. We define the remaining operations by the ones that have not begun by the time t_0. This implies that the operations which are in progress at time t_0 should be completed first in order to identify the earliest availability of each machine.

3.2.1 Full Rescheduling method

The following steps define how *full rescheduling* is performed with respect to each type of disturbance:

Step 1: initialization

Input the initial schedule generated by FBS (operations, assignment, starting time), input the value of filterwidth (f) and beamwidth (b), and input the detailed information of machines, jobs and operations.

Step 2: Disturbance Occurrence-(Re-initialization)

At time t_0=TNow, the system need to be rescheduled to respond to a disturbance. Depending on the type of disturbance, different scenarios may stand out;

Case 1: Machine Breakdown- In this study, a predetermined repair time has been considered for each machine. It is also assumed that machines would be available after repair time to be loaded and non-preemptive is allowed.

In case of machine breakdown at time t_0=TNow, the operations in progress on the other machine should be completed to identify their availabilities. If the failed machine has an operation in progress, the remaining part of that operation should be served by that machine after repair time. Starting time of operations and machine availabilities are updated and the system is ready to be scheduled for the remaining operations.

Case 2: Job Cancellation – If a job is cancelled at time t_0=TNow, all its remaining operations would be cancelled even if it is in progress. Then, the initial partial schedule is formed by all the operations have been finished by TNow or in progress in TNow and not-cancelled. The starting time and machine availability is updated and partial schedule is ready to be added by remaining operations.

Case 3: New Order arrival –If a new order arrives to system at t_0=TNow, first, the in-progress operations will be finished (non-preemptive), second, the new order is included into the list of remaining operations and initial partial schedule is formed by all operations have been performed before TNow, in progress in TNow. The starting time and machine availability is updated and partial schedule is ready to be added by remaining operations.

Step 3: Generating nodes

(i) By employing M_Nondelay method, generate nodes from the updated existing partial schedule. Check the total number of nodes generated, N, update level and the partial schedule PS_l by generating nodes.

(ii) If $N< b$, then go down to next level, generate new nodes by M_Nondelay and PS_l, update level and PS_l by generating nodes. If $N< b$, then go to Step 3. (ii); else go to Step3.(iii).

(iii) Find the global evaluation function values for all the nodes and select the best b number of nodes (defining the initial beam nodes). Determine the candidate sets of each beam $PS_l(1)$, $PS_l(2)$,…, $PS_l(b)$

Step 4: Determining beam nodes

Check the level and number of operations remained. If any operation remained, go to step 4.(i); else go to Step 5.

(i) Generate N new nodes from each beam node according to M_Nondelay with PS_l as the partial schedule represented by the beam node. If N<b go to Step 4.(i); else go to Step 4.(ii).

(ii) Filtering process. Choose the best f number of nodes according to local evaluation function values.

(iii) Computing global evaluation functions. The global evaluation function for each filtered nodes are computed.

(iv) Selecting the beam nodes. Select the nodes with the lowest global function and update the partial schedules and level.

Step 5: select the solution schedule

Among the beamwidth number of generated schedules, select the schedules set or schedules sets with the best objective function values.

3.2.2 Do nothing method

The following steps define how the *do nothing* method is performed with respect to each type of disturbance:

Case 1: Machine Breakdown– in case of machine breakdown at t_0=TNow, all the remaining operations on the failed machine will be right shifted and start after the repair time. This shifting may also cause delaying the operations that are allocated on other machines due to the precedence constraints.

Case 2: Job Cancellation – in case of job cancellation at t_0=TNow, all the remaining operations will be left shifted to an earlier starting time on their current assigned machine while considering precedence constraints.

Case 3: New Order arrival –in case of a new order arrival at t_0=TNow, the operations of new jobs will be allocated by SPT into the first available machine. If no machine is available, they will be added to the end of schedule as a result.

3.3 Proposed Objective function

3.3.1 Rescheduling cost

Three main rescheduling cost sources distinguished in this study can be categorized as machine related, job related, and material related cost:

$$OBJ(Cost) = Cost_{\mathrm{Re}sch} = Cost(Mach)_{\mathrm{Re}sch} + Cost(job)_{\mathrm{Re}sch} + Cost(Mat)_{\mathrm{Re}sch} \qquad (1)$$

- Machine related Cost:

The idle time of machinery in FMS is a key factor to be considered. In addition, due to schedule update, an extra machining cost can result from switching an operation to an alternative machine with longer processing time. Then, the machine related rescheduling cost is the total cost of increased the idle time of all machines and total cost of extra processing time after rescheduling which are given by the following equations:

$$Cost(Idle\ time)_{\mathrm{Resch}} = \sum_k Max[(Makespan_{\mathrm{Re}sch} - \sum_k P_{ijk})_{\mathrm{Re}sch} - (Makespan_{initial} - \sum_k P_{ijk})_{initial}, 0] \times \eta_k \qquad (2)$$

$$Cost(Extra\ machining) = Max[(\sum_j \sum_i P_{ijk})_{\mathrm{Re}sch} - (\sum_j \sum_i P_{ijk})_{Initial}, 0] \times \tau \qquad (3)$$

The coefficient η_k corresponds to the rate of idle time for each machine. P_{ijk} Represents the processing times of operation *i* of job *j* which performs on machine *k*. τ represent the penalty cost of extra machining process needed after rescheduling. The first term in Equation (2) corresponds to total idle time of machines in updated schedule and second term corresponds to total idle time of machines in initial one. In Equation (3), terms identify total machining works in updated and initial schedule respectively.

- Job related Cost:

By running rescheduling, jobs could be shifted and finish earlier or later than the initial schedule. The associated cost is the cost of added lateness of jobs after rescheduling which can be expressed by the following equation:

$$\sum_j Max[Max(C_{ij} - D_j, 0)_{\mathrm{Re}sch} - Max(C_{ij} - D_j, 0)_{Initial}, 0] \times \delta_j \qquad i = n_j, \forall\ i, j \tag{4}$$

The coefficient δ_j represents the penalty cost of unit time lateness for each job. C_{ij} is the completion time of operation *i* of job *j* and D_j is the job due date. The first term in Equation (4) is the tardiness of each job in updated schedule while the second term corresponds to the tardiness of jobs in initial schedule.

- Material related Cost:

Once we generate the initial schedule, it is assumed that the required raw material for each operation is supplied just before the start time of operation, then changing the start time or machine assignment may incur a cost. The material related cost consists of three types of cost functions such as holding cost of WIP and raw material, cost of expediting the material, and cost of reallocating the material to another machine. Since we perform full rescheduling, the starting time of operations and assigned machines are subject to change in the new schedule. If the rescheduled operation starts later, the raw material should be kept to be used later which cause holding cost. Also, if the rescheduled operation has to start earlier than the original schedule, the raw material should be ordered sooner and extra fee should be paid for expediting. Operations may also be assigned to different machines after rescheduling; and this change causes the cost of reallocation such as changing toolsets, and extra material handling. The corresponding cost functions are given as follows:

$$\forall k;\ S'_{ijk} > S_{ijk} \qquad \text{Cost(H)} = \sum_{i}\sum_{j}(S'_{ijk} - S_{ijk}) \times h_{ij} \tag{5}$$

$$\text{if } \forall k;\ S'_{ijk} < S_{ijk} \qquad \text{Cost(Expd)} = \sum_{i}\sum_{j}(S_{ijk} - S'_{ijk}) \times \mu_{ij} \tag{6}$$

$$\forall i, j, k \qquad \text{Cost(Raloc)} = \sum_{i}\sum_{j} Y_{ijkk'}\omega_{ijkk'} \tag{7}$$

The coefficient h_{ij} and μ_{ij} represent the holding and expediting cost of material for the operation *i* of job *j* in unit of time. S'_{ijk} is the start time of operation *i* of job *j* on machine *k* after rescheduling and S_{ijk} is the original start time of that operation. $\omega_{ijkk'}$ represents the penalty cost of reallocating material between machines after rescheduling process. $Y_{ijkk'}$ is the binary variable indicating whether the operation *i* of job *j* on machine *k* is switched to machine *k'* or not.

3.3.2 Potential and Actual cost of rescheduling

The proposed rescheduling cost function evaluates the cost impact assuming that the remainder of the schedule will not change. Since there can be further disturbances during the execution of an updated schedule, this may result in further updates in the starting time and reallocation of the operations that have not yet begun. For example, if an operation is right shifted at $t_0=2$ from 4 to

8 in case of machine breakdown, then it may be left shifted due to a job cancellation at $t_1=4$ from 8 to 6. The actual cost of rescheduling actions can be calculated as holding cost of 2 time units. Then, the actual total cost of rescheduling actions will not be equal to the sum of rescheduling costs found at each change. Thus, the cost which is calculated in each step of disturbance may be considered as a surrogate measure for total rescheduling cost. The decision is made based on this potential cost at each level and the actual cost of action can be derived at the end of planning horizon by looking at the final schedule and compare it to initial schedule considering all changes have been performed during the execution of the schedule.

4. Computational results

In order to demonstrate the efficiency and performance of the proposed algorithm for rescheduling in the FMS environments, a numerical study is developed, tested and evaluated. The algorithm is run on a personal computer with an Intel Core 2 Duo CPU, 2 GB RAM on Microsoft Windows XP Professional. The codes are written in the Java, Eclipse (Galileo 3.5.1 platform).

The test problem considered in this study is a FMS system with 4 partially flexible machines, with 4 jobs having 4 operations on each job. The required parameters are defined as follows:

η_k represents machine idle time cost. It is assumed that idle time cost is equal for all machines and is set to be 6 \$/unit of time. δ_j, lateness cost for each job, is assumed to be equal for each job and is set to be 12 \$/unit of time. It is also assumed that batch size for all operations are equal, then the holding cost, h_{ij} ,would be equal for all operations and is set to be 1\$/unit of time. Similarly μ_{ij} is set to be 10 \$/unit of time, $\omega_{ijkk'}$, reallocation cost, is set to be \$4 for all operations and finally τ as penalty cost of extra machining, is set to be 6 \$/unit of time. Repair time for machine 1, 2, 3 and 4 are 5,4,4,3 respectively.

4.1 Initial Schedule generation

In order to generate the initial and updated schedules with an acceptable performance level and computational time, we need to determine the proper value of *filterwidth (f)* and *beamwidth (b)* of the FBS algorithm. In this case study, the appropriate values of these parameters are obtained via experimental trials. *b* is set to be between 2 and 9 and *f* is set to be between 2 and 6. According to the results shown in Table1, setting *f*=4 and *b*=3 gives the best compromise between the optimal result and the size of the search space.

	***F* Value**				
	***f*=2**	***f*=3**	***f*=4**	***f*=5**	***f*=6**
***b*=2**	21.7	21.7	21.7	21.7	20.5
***b*=3**	20.3	20.3	18.9	18.9	18.9
***b*=4**	21.3	19.2	18.9	18.9	18.9
***b*=5**	20.3	19.2	18.9	18.9	18.9
***b*=6**	21.3	19.2	18.9	18.9	18.9
***b*=7**	21.3	19.2	18.9	18.9	18.9
***b*=8**	21.3	19.2	18.9	18.9	18.9
***b*=9**	21.3	19.2	18.9	18.9	18.9

Table1. Objective function value for different *filterwidth* and *beamwidth*

The initial schedule, generated by using makespan criteria, is represented by the following Gantt chart.

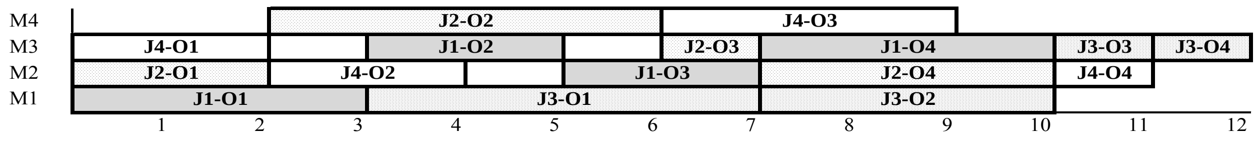

Figure 2: The static schedule result for a FMS problem with *b*=3, *f*=4, *F*=18.9

This function prepares a good balance between makespan, machine workload and total processing time and the coefficients of each term are considered to be 0.4, 0.3 and 0.3 respectively [6].

4.2 Rescheduling example for machine breakdown

The machines which are failed to perform their operations are selected randomly through the program. In the Figure 3, machine 4 is failed at time t=4 for ΔT=3 as a repair time. The rescheduling cost can be calculated by observing the differences of the initial and updated schedules represented in Figure 2 and Figure 3:

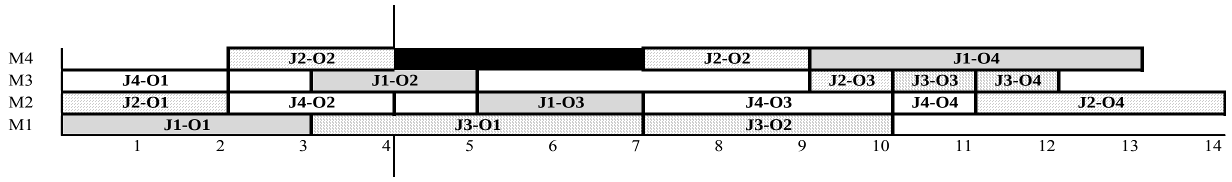

Figure 3: Gantt chart obtained after rescheduling, in case of machine 4 failure at t=4

4.3 Do nothing example for a new order arrival

The jobs which are arrived to the system are selected randomly through the program. In Figure 4, job number 5 arrives to the system at time t=7. The rescheduling cost can be calculated by observing the differences of the initial and updated schedules represented in Figure 2 and 4.

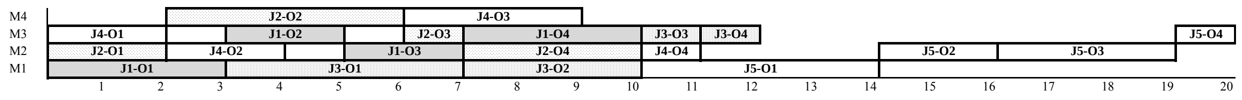

Figure 4: Gantt chart obtained after adding new order

4.4 Performance analysis

To check the performance of the proposed methodology, 5 different scenarios are developed in this study each having different numbers and types of disturbances affecting the existing schedule. For example, in case of 7 disturbances during the horizon following disturbances occur:

- at time t=2 a new order arrives to the system,
- at time t=3 job number one is cancelled,
- at time t=5 a new order arrives to the system,
- at time t=8 machine number two is failed for ΔT=4,
- at time t=9 job number four is cancelled,
- at time t=12 another job arrives to the system,
- And at t=14 machine one is failed for ΔT=4.

The types and the time of disturbances are randomly defined. The result of this scenario is represented in Table2. As shown in the table, at time t=2, the cost of *do nothing* is less than the cost of *full rescheduling,* then the initial schedule is replaced with updated one with *do nothing* method. In next disturbance, t=3, the cost of *fully rescheduling* is less than *do nothing* policy and this one would replace to the existing one. The total potential cost is the summation of costs in each step and actual cost is calculated at the end of horizon by comparing the final schedule with the initial one and the analysis of the actual changes performed during the execution of the updated schedules.

Policy	**Time of Disturbances**							**Potential Cost**	**Actual cost**
	t=2	t=3	t=5	t=8	t=9	t=12	t=14		
Do Nothing VS Fully Rescheduling	**DN(144)** 177	140 **FR(94)**	168 **FR(58)**	**DN(78)** 78	**DN(32)** 58	168 **FR(144)**	116 **FR(82)**	**632**	**400**
Fully Rescheduling	177	120	123	91	38	144	82	**775**	**420**

Table 2: comparing the costs of the proposed methodology and always fully rescheduling

In Figure 5, the potential cost of do nothing and fully rescheduling is represented in a graph. At each time of disturbance the lower point identifies the better policy to proceed.

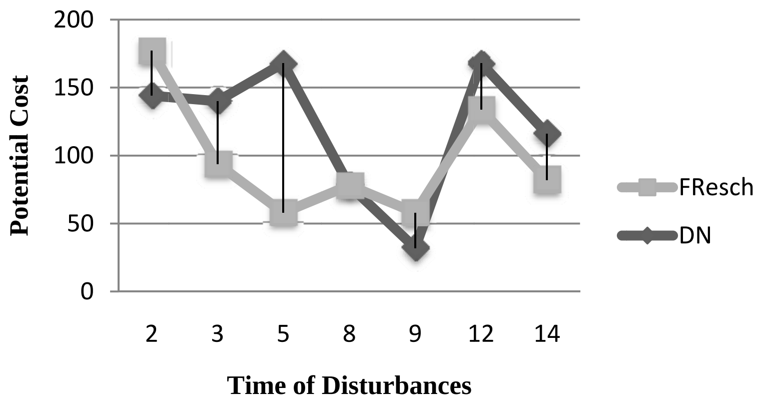

Figure 5: comparing the cost of fully rescheduling and do nothing in the proposed methodology

In Table 3, the potential and actual cost of action of two methodologies for 5 different cases is represented. *P-Method* represents the proposed method in this study and *F-Resch* represents the case of responding to each disturbance by fully rescheduling.

Number of disturbances	**3**		**4**		**5**		**6**		**7**	
Methodology	P-Method	F-Resch	P-Method	F-Resch	P-Method	F-Resch	P-Method	F-Resch	P-Method	F-Resch
Potential Cost	258	384	314	352	516	481	550	693	632	775
Actual Cost	164	271	265	274	367	382	388	392	400	420

Table 3: potential cost and actual cost of different methodologies for different number of disturbances

4.5 Discussion of results:

The computational results shown in Table3 compares the performance of the proposed rescheduling method with *full rescheduling* the system in each disturbance. The actual cost of rescheduling actions resulting from the *P-method* is less than the actual cost of *F-Resch* method in all considered scenarios. This means that rescheduling the system for every disturbance creates more cost of actions. In addition, more frequent rescheduling may cause increased nervousness of the system. The proposed rescheduling cost function helps generating schedule updates with minimum possible costs. The results also show that there are better ways of rescheduling as opposed to selecting one type of rescheduling methodology since the schedule is somewhat repaired by right or left shifting in *do nothing* case. As expected, the cumulative sum of potential rescheduling costs is not equal the actual cost of rescheduling actions. However, it can be used as a surrogate measure to determine the type of rescheduling methodology for incoming disturbances and generate schedule updates while minimizing the cost of updates.

5. Conclusions

This work has attempted to address a new practical method in rescheduling problem of flexible manufacturing systems. For this purpose, filtered beam search algorithm is applied in order to generate schedules with the proposed cost function. A decision methodology is developed to find less costly way of responding to disturbances and the results were verified by evaluating the total actual cost of rescheduling actions. Throughout this study, the cost calculations are done by only considering the incoming disturbance. Future research will include developing an expected rescheduling cost function which considers the arrival of additional disturbances for the remainder of the planning horizon.

References

1. Vieira, G.E., Hermann, J.W., Lin, E., 2003, “Rescheduling manufacturing systems: a framework of strategies, policies and methods” Journal of Scheduling, 6(1), 39-62.
2. Sabuncuoglu, I., Karabuk, S., 1999, “Rescheduling Frequency in an FMS with Uncertain Processing Times and Machine Reliabilities”, Journal of Manufacturing Systems, 18(4), 268-283
3. Abumaizar, R.J., Svestka, J.A., 1997, “Rescheduling job shop under random disruptions” International Journal of Production Research, 35(7), 2065-2082.
4. Wang, S.J., Zhou, B.H., Xi, L.F., 2007, “Filtered-beam-search-based algorithm for dynamic rescheduling in FMS” Robotics and Computer-Integrated Manufacturing, 23(4), 457-468
5. Sabuncuoglu, I., Goren, S., ”Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research”,
6. Wang, S.J., Zhou, B.H., Xi, L.F., 2008, “A filtered-beam-search-based heuristic algorithm for flexible job-shop scheduling problem” International Journal of Production Research, 4(11), 3027-3058.

7. Baker, K.R., 1974, “Introduction to sequencing and scheduling” Wiley, New York.
8. Hoitomt, D.J., Luh, P.B., Patttipati, K.R., 1993, “A practical approach to Job shop scheduling problems” IEEE Transactions on Robotics and Automations, 9(1), 1-13.
9. Goren, S., Sabuncuoglu, I., 2008, “Robustness and stability measures for scheduling: single-machine environment” IIE Transactions, 40(1), 66-83.
10. Sabuncuoglu I, Bayiz M. “Analysis of reactive scheduling problems in a job shop environment” European Journal of Operation Research, 2000(126), 567-86.
11. Ow PS, Morton TE. “Filtered beam search in scheduling” International Journal of Production Research, 1998; 36(9), 2609-26.