An Improved Genetic Algorithm for Multi-objective Dynamic Scheduling Optimization

Kunlei Lian[1], Chaoyong Zhang[1], Liang Gao[1], and Chaoyang Zhang[1]

[1]State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan, China 430074

Abstract

Dynamic scheduling problem is a more complex NP-hard problem compared with static scheduling problem and in most cases it has multi-objective performance criteria. Based on study of the multi-objective dynamic scheduling problem, it employs the rolling-horizon procedure and an improved genetic algorithm. In the rolling-horizon procedure, dynamic scheduling problem is decomposed into a series of continual and static scheduling problems, multi-objective genetic algorithm is applied to each of these problems. In order to adapt to the complex manufacturing environment and sustain the stability of production, a human-compute collaborative scheduling procedure is presented for the implementation of the scheduling process. A scheduling prototype system is developed and tested on the improved job-shop benchmark instance, the simulation results validate the effectiveness of the proposed strategies.

Key words: Job shop scheduling problem; dynamic scheduling, genetic algorithm; rolling horizon

Jackson first proposed the concept of dynamic scheduling in 1957 considering the difficulty of applying traditional static scheduling models in real-world manufacturing environments that are characterized by dynamism, randomness and multiple objectives. The proposal of dynamic scheduling models is mainly due to frequent occurrences of disturbed schedules inflicted by a series of stochastic impacting factors encompassing new order arrivals, part scrap and rework, due date changes, machine malfunction or delayed raw material arrivals. Rescheduling is therefore necessary in order to resume manufacturing process based on the updated part status in the system.

Traditional integer programming methods can be hardly used in solving real-world dynamic scheduling problems. On the other hand, advancements in computer technologies open new opportunities in solving dynamic scheduling problems using artificial intelligence, simulation, rolling-horizon rescheduling and meta-heuristics, which lays the foundation for practical usage of manufacturing scheduling models. Artificial

intelligence and expert systems can identify the best scheduling strategy by searching knowledge databases based on current system status and predefined optimization objective. However, they suffer from poor adaptivity to new environment, high development cost and prolonged development cycle. Discrete system simulation method simulates real-world manufacturing environment by creating simulation models, but general principles are difficult to find due to its experimental nature. Rolling horizon rescheduling was original proposed by Nelson in 1977 and has received numerous research attentions and applications in recent years. It divides the dynamic scheduling process into multiple continuous scheduling horizons and then conducts online optimization for each horizon in order to find the individual optimal solution, which makes it applicable to complex dynamic manufacturing scheduling environments (**??**).

Genetic algorithm, one of the meta-heuristic algorithms, has witnessed many applications in dynamic scheduling researches due to its simple operations, high efficiency, good robustness, superior adaptability and intriguing searching capability based on individual fitness. This paper proposes a hybrid algorithmic framework of multi-objective genetic algorithm and rolling horizon rescheduling to solve the real-world dynamic scheduling problem considering possible delays of raw material arrival, part machining times and assembly times. It can also tackle the emergent insertion of parts and continuous arrivals of planned jobs. This framework can be used in other manufacturing scenarios as well.

1 Dynamic Scheduling Problem Description

Static scheduling problems involve n jobs to be processed on m machines and the scheduling plan can be determined after the processing order is decided for every job on all the machines. However, in real-world manufacturing systems, this processing order needs to be rescheduled whenever new orders arrive, machines break down or raw materials get delayed.

The part machining in manufacturing environments is viewed as a dynamic process where jobs become available for processing sequentially, followed by machine processing and exiting the manufacturing system after their processes are finished. Dynamic events refer to entities that trigger the rescheduling of existing jobs due to changes in scheduling environment, and they can be classified into four categories:

- Job-related events, these include stochastic arrivals of jobs, randomness in job processing times, changes in delivery due date, dynamism in job processing priorities and and order changes.
- Machine-related events, these include machine break-downs, limited capacity, machine deadlock and conflicts in manufacturing capacity.
- Process-related events, these include process delay, quality negation and output un-stability.

- Other events, these include absent operators, delayed arrival or unexpected flaws of raw materials, and dynamic processing routes.

In the classical job shop static scheduling problems, the release time r_i of every job is assumed to be zero and the objective is to minimize the make-span C_{max}, which is defined as the maximum completion time of all jobs, $C_{max} = \min\{\max C_i,\ i = 1, \cdots n\}$ where C_i is the completion time of job J_i. In real-world dynamic manufacturing environments, however, jobs become available for process sequentially, meaning that their release times r_i are different and unpredictable. Since a job can only be processed after it becomes available, the maximum completion time of all jobs in dynamic scheduling problems is determined by the completion time of the latest-released job. Therefore, dynamic scheduling problems generally use the mean flow time of all jobs, $\bar{F}$, as the objective function instead of the maximum completion time of all jobs.

This paper considers two performance metrics that are often seen in dynamic manufacturing environments: 1) minimization of mean flow time $\bar{F}$ where $\bar{F} = \min(\frac{1}{n} \times \sum_{i=1}^{n} C_i - r_i)$ and r_i and C_i are the release time and completion time of job J_i, respectively. 2) minimization of the weighted objective of maximum completion time C_{max} and total tardiness. This weighted objective gives higher priority to urgent jobs which are required to finish by their delivery due dates, and minimizes the completion times of remaining jobs. For a scheduling problem with n jobs and m machines, the objective can be defined as follows:

$$\min(\max C_i + \alpha \times (\sum \max(0, C_j - D_j))),\ (i \in S_{J_1}, j \in S_{J_2}) \tag{1}$$

where α is the weighted penalty coefficient, S_{J_1} is the job set that is being processed and S_{J_2} is the newly inserted urgent job set, C_i is the completion time for job $i \in S_{J_1}$, and D_j is the delivery date for urgent job $j \in S_{J_2}$, and C_j is the completion time.

2 Rolling Scheduling Strategy

The real-world manufacturing system scheduling is undeterministic due to unpredictable events or stochastic disturbances. Raman (**?**) and Jian(**?**) proposed a rolling horizon strategy to turn the undeterministic scheduling problem into a series of dynamic but deterministic scheduling problems. This is based on the successful industry applications of utilizing predictable control in continuous sytems to replace optimal control. The scheduling process is divided into continuous staitc scheduling horizons and each one is solved sequentially. This strategy is able to address the impact of undeterminisitc factors as well as incorporate system changes into scheduling plans.

Rolling optimization is the key to rolling scheduling, which conducts static job scheduling by first removing

finished jobs from the current scheduling horizon and then including available new jobs into the current scheduling horizon. This process is repeated until all jobs are finished processing. The selection of rolling horizon and job selection strategy play key roles in the overall scheduling efficiency.

Applying rolling horizon into scheudling problem requires defining a rolling hoziron which encompasses multiple job sets: finished job set, processing job set, un-processed job set, available job set. In every rolling scheduling step, finished job set is first removed from the current rolling horizon, and available job set is then added, followed by statis scheduling optimiztion of this new job set.

The rescheduling cycle is defined as the time interval of two consecutive schedulings. Normally, rescheduling times are evenly distributed, which does not consider the workload status of real-world manufacturing systems. A more resonable strategy is to associate rescheduling frequency with manufacturing workload. The number of jobs to be scheduled is limited by two factors: 1) more jobs are desired in order to improve machine utilization; 2) less jobs are preferred if the system has to incorporate urgent job insertions. The decision is largely based on real-time circumstances.

There are three types of rolling-reschedulings: event-driven rescheduling, cycled rescheduling and hybrid rescheduling of the two. Event-driven rescheduling refers to a rescheduling triggerred by the advent of system status-changing events, which may include delayed arrivals of raw materials, delayed processing and machine break-downs. Cycled rescheduling refers to a rescheduling strategy controlled by pre-defined time intervals, which can be decided by planned deliveries, manufacturing workloads. There two strategies are not able to incorporate future events and cycled rescheduling cannot tackled urgent events(**?**). The hybrid strategy is more responsive to real-world dynamic manufacturing environments and therefore more stable. This paper employs this hybrid strategy by using cycled rescheduling as the general strategy and switching to event-driven rescheduling when urgent jobs emerge, including delayed raw materials arrive, machines break down, job delivery dates change and urgent jobs arrive.

3 Dynamic Scheduling Strategy

This paper proposes an improved genetic algorithm within the rolling horizon framework utilizing the hybrid cycled and event-driven rescheduling strategy. The job rescheduling within a horizon is achived by the proposed genetic algorithm and the solution can be used for exection. The following sections introduce the main components of the dynamic scheduling problems.

3.1 Dynamic scheduling solution encoding scheme

This paper uses a process-based encoding scheme in which all the processes of a job are indicated by the job number, and the kth occurance of the job indicates the kth process of the job.

3.2 Dynamic scheduling solution decoding scheme

The scheduling solutino decoding scheme refers to the determination of the starting times of all process on all machines based on the current solution chromosome and process plans. For a process i, given a starting time t_i and processing time of p_i, its completion time can be computed as $(t_i + p_i)$. The preceeding process of process i is denoted by $JP[i]$ and the preceeding process of machine is $MP[i]$ if it exists. In dynamic scheduling problems, the machining starting times of jobs that are being processed or un-processing, and not the first process, can be calculated as

$$t_i = \max(t_{JP[i]} + p_{JP[i]}, t_{MP[i]} + p_{MP[i]}) \tag{2}$$

the starting times of the first process can be computed as

$$t_i = \max(\max(t_{JP[i]} + p_{JP[i]}), t_{MP[i]} + p_{MP[i]}) \tag{3}$$

For available processes, the starting time t_i of the first process can be computed by

$$t_i = max(r_i, t_{MP[i]} + p_{MP[i00]}) \tag{4}$$

This paper uses a decoding algorithm based on greedy insertion and can gurantee to produce a feasible schedule after solution decoding.

3.3 Crossover and mutation operators

There exist many crossver operators in the literature, including PPX**?**, SPX(**?**), POX(**?**), among which POX is able to inherit promising characteristics of parent solutions and is depicted in figure **??**.

The mutation operator randomly selects a gene and inserts it into another position in order to produce a small solution perturbation.

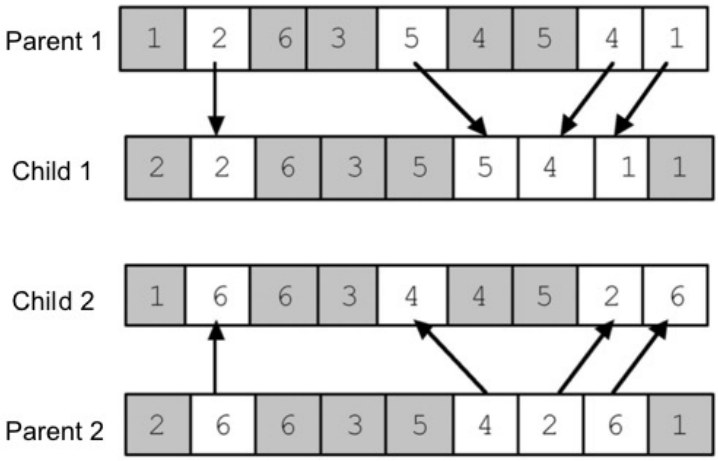

Figure 1: POX crossover operator based on job sequence encoding

3.4 Selection operartor

Selection operator decides whether an individual enters the next generation based on its fitness value, which is defined as the objective function. The selection operator in genetic algorithm uses both strategies of elite model and tournament selection. The former identifies the best 1% individuals in the current generation and allows them entering into the next generation directly without crossover. Tournament selection works by randomly selecting two individuals from the current generation and then chooses the one with better fitness value if a random number $r \in (0, 1)$ is smaller than a predefined value k which is set as 0.8. Note that the chosen one will be allowed to be selected again in the next iteration of tournament selection.

3.5 Improved genetic algorithm design

Traditional genetic algorithm does not perform well in solving scheduling problems due to the acceptance of new offsprings created though the crossover operator, even if their fitness values are worse than that of their parent solutions, during which process good solutions are lost and damaged. This paper proposes an improved genetic algorithm utilizing a new offspring solution creation strategy. In this strategy, $2n$ offspring solutions are created from n iterations of crossover operators applied on two parent solutions, the two best offspring solutions are chosen as the final offspring solutions to enter the next iteration. Computational results show that the improved algorithm performs superior than traditional genetic algorithm in both convergence rate and solution quality. For example, the optimal solution with objective value of 930 is obtained using the improved genetic algorithm in solving the famous benchmark instance FT10. The algorithmic workflow is as follows:

1. randomly create P chromosome individual solutions and compute their corresponding objective values.
2. output the optimal solution if the stopping criteria is met; go to the next stop otherwise.
3. select the offspring solutions for the next generation according to the selection operator.
4. (1) apply crossover operator with probability P_c on parent solution n times to create $2n$ offspring solutions, from which two best solutions are selected as the final offspring solutions; (2) apply mutation probability P_m on offspring solutions to create mutated offsprings.
5. create new population and return to step 2.

4 Computational Experiments

We developed a prototype scheduling system based on the described improved genetic algorithm and validated its performance based on testing instances. Since there exist no benchmark instances in the literature specifically designed for dynamic scheduling problems, we create testing instances based on the widely used FT20 problem and add dynamic data when dynamic scheduling is required. These testing instances are then used to simulate the scheduling problems in dynamic manufacturing environments where unexpected events and urgent jobs arrival happen often. In the experiment, minimization of mean flow time is used as the objective function in the case of unexpected events; a multi-objective of maximal makespan and total tardiness is used as the objective function in the case of urgent job insertion.

Parameters of the improved genetic algorithm are set as follows: population size $P = 200$; crossover probability $P_c = 0.8$; crossover times $n = 10$; mutation probability $P_m = 0.01$ and total number of iterations as 100. Figure **??** shows the initial scheduling plan for the FT20 instance with 20 jobs to be processed on 5 machines. This initial plan represents the optimal solution for this problem and the maximum completion time is 1165.

Suppose the following stochastic events happpen on time 600: 1) processing time of job $(J_{10}, 1)$ on machine 2 is delayed for 30 units of time, due to machine breakdown, work delay or other reasons; 2) jobs $(J_1, 3)$ and $(J_{20}, 3)$ on machine 3 are exchanged due to worker absenteeism or vacation; 3) starting time of job $(J_8, 1)$ on machine 3 is delayed for 60 time units due to delayed arrival of raw material; 4) starting time of job $(J_1 1, 4)$ on machine 5 is delayed for 40 time units due to delayed assembly; The maximal completion time is delayed to 1219 after the manual adjustments to accomodate the above stochastic events. The improved genetic algorithm is applied at time 600 to reschedule refreshed jobs and figure **??** shows the optimal schedule of this scenario and the maximal completion time is reduced to 1202.

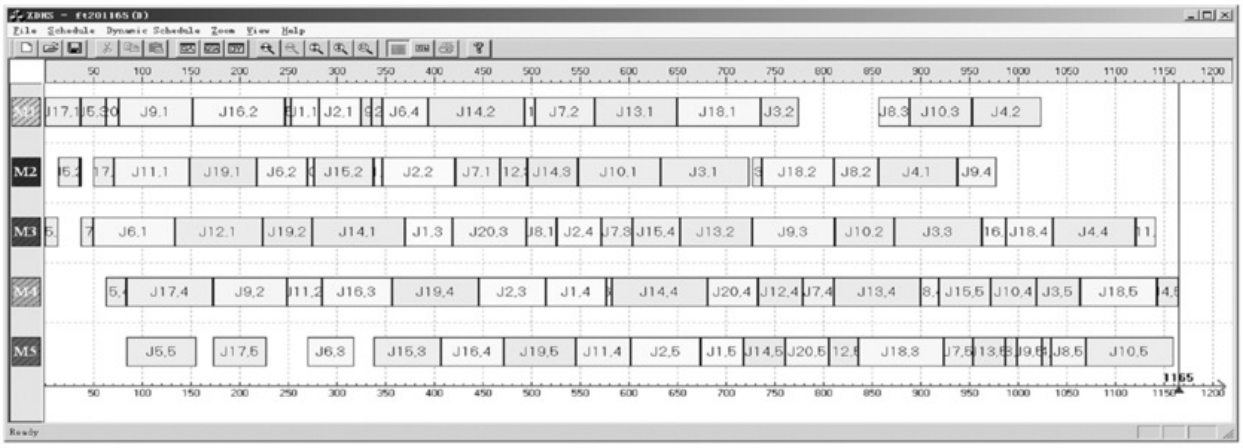

Figure 2: Initial scheduling plan

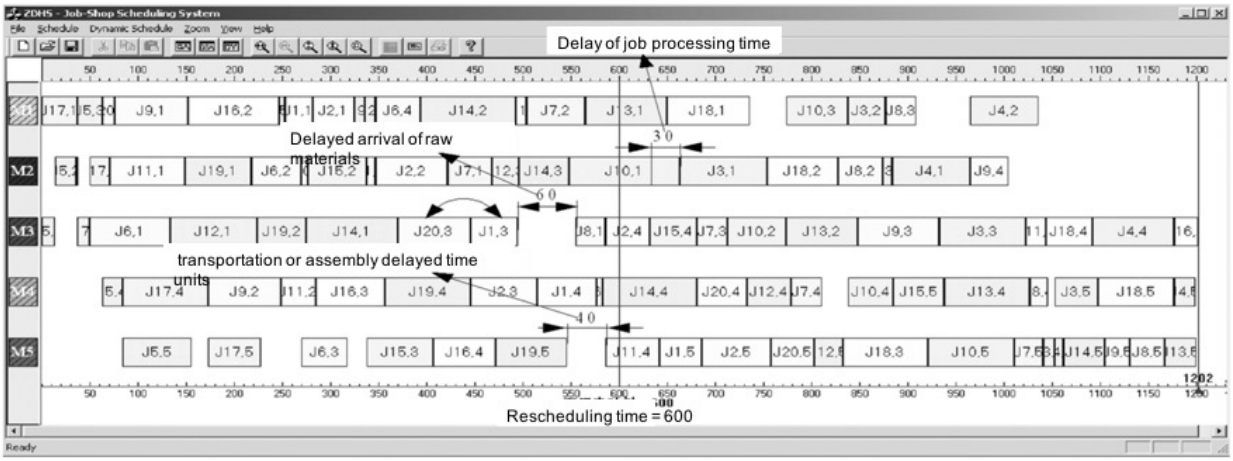

Figure 3: Event-driven rescheduling plan

To futher validate the performance of rolling horizon optimization, we assume the arrival of urgent jobs at time 600 in addition to the aforementioned stochastic events. There are three jobs in the urgent arrival, namely, job 21, 22 and 23, and they are required to finish before given delivery date. Table **??** gives the processing machines, processing times, release times and delivery dates of the three jobs. Note that delivery date is defined by multipling average processing time by 1.8 and adding the result to the rescheduling time point of 600. In this scenario, the scheduled jobs need to be processed as sson as possible, whicle new urgent jobs must be finished before delivery date. To this purpose, we need to solve the multi-objective dynamic scheduling problem considering both maximal job completion time and total tardiness. Using this multi-objective function, the improved genetic algorithm is used to reschedule all the jobs at time 600 and figure **??** shows the optimal solution. It can be seen from the figure that all the three newly inserted jobs are finished processing before time 900 and the maximal completion time is 1388.

Table 1: Processing machine, processing time and release time for each urgent job

job	processing machine/time					release time	delivery due time
	1	2	3	4	5		
job 21	1(29)	2(9)	3(49)	4(12)	5(26)	650	900
job 22	3(18)	2(22)	1(36)	4(21)	5(72)	600	900
job 23	2(46)	1(28)	3(32)	5(40)	4(30)	680	900

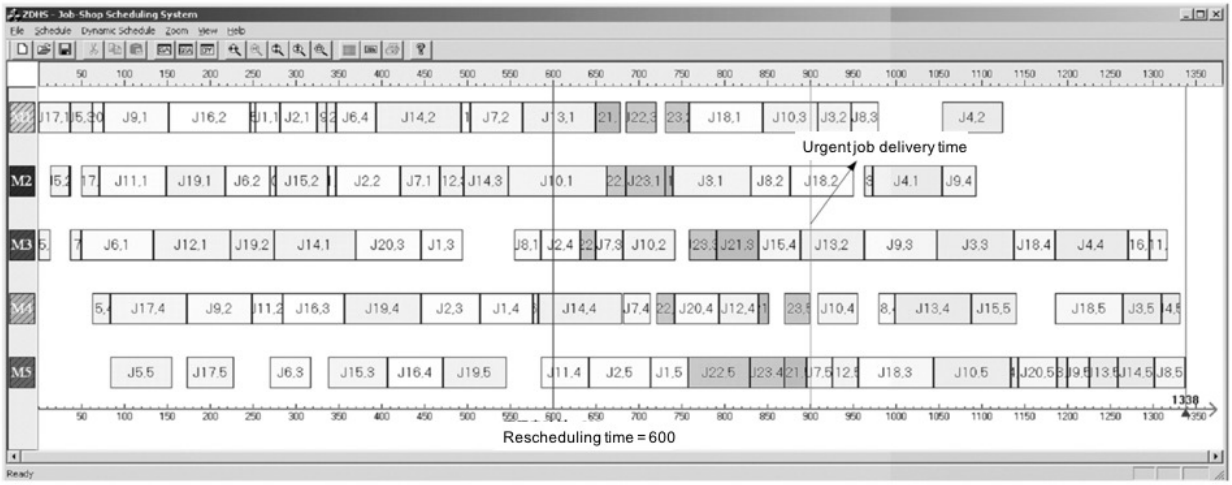

Figure 4: Multi-objective dynamic scheduling plan after insertion of urgent jobs

The above rescheduling strategy can be used together with the improved genetic algorithm to continuously optimize all the jobs within a rolling horizon. The computational results show that a complex dynamic scheduling problem can be naturally divided into multiple scheduling horizons and static scheduling algorithm can be used to optimize job schedules within each horizon, which improves the scheduling capability in adapting to dynamicly changing manufacturing environmnets and urgent events and jobs can be tackled promptly. This strategy can be used in other dynamic scheduling problems as well.

5 Conclusions

This paper studies the multi-objective job shop scheduling problems in dynamic manufacturing environments by fully utilizing the static scheduling capabilities of algorithms in the literature. We propse a rolling horizon rescheduling strategy based on an improved genetic algorithm to tackle stochastic events, including unexpected events, urgent job arrivals and continuous arrival of planned jobs. The proposed genetic algorithm is able to conduct online optimization within each rescheduling horizon and make the rolling horizon startegy adaptable to complex dynamic scheduling environment changes. A prototype dynamic scheduling system is developed based on the rolling horizon rescheduling strategy and comptuational experiments validated its

feasibility and effectiveness.