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, 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 emploies the rolling-horizon procedure and an iimproved genetic algorithm. In the rolling-horizon procedure, dynamic scheduling problem is decomposed ino a series of continual and static schedulig problems, multi-objective genetic algorithm is applied to each of tehse 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 effectivenss of the proposed strategies.

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

Traditional static scheduling models cannot be used in real-world manufacturing environment with dynamic, stochastic and multi-objective nature, and the concept of dynamic scheduling was first proposed by Jackson in 1957. The advent of dynamic scheduling models is due to the disturbed schedules inflicted by a series of stochastic impacting factors emcompassing the arrival of new orders, scrap and rework of certain parts, changed due date, machine malfunction or delayed raw material arrival. Rescheduling is therefore necessary in order to resume manufacturing process based on the current part status in the system.

Traditional integer programming methods can be hardly used in solving real-world dynamic scheduling problems. On the other hand, new advancemens in computer technologies open new opportunities in sovling dynamic scheduling problems using artificial intelligence, simulation, rolling-horizon rescheduling and meta-heuristics, which lays the foundations for practical usage of manufacturing scheduling models. Artificial intelligence and expert systems can identify the best schedulig strategy by searching knowledge database 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, from which general principles are difficult to find due to its experimental nature. Rolling horizon rescheudling was original proposed by Nelson in 1977 and has received numerous research attentions and applications in recent years. It divides the dynamic scheduling problem into continual scheduling sections and conducts on-line optimization for each section in order to find the individual optimal solution, which makes it applicable to complex dynamic manufacutring scheduling environments.

Genetic algorithm, one of the metaheuristic algorithms, has witnessed many applications in dynamic scheduling researches due to its simple operations, higher efficiency, better 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 delayed raw material arrival, part maching time and assembly time. It can also tackle the emergent insertion of parts and continuous arrival of scheduling 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.

Dynamic scheduling views the job manufacturing as a dynamic process in which jobs become available for processing continuously, 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, undetermined job processing times, changes in delivery date, dynamic priorities and orders.
- machine-related evenst, these include machine break-down, limited capacity, machine deadlock and conflicts in manufacturing capacity.
- process-related events, these include process delay, quality negation and output unstability.

- 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 times r_i of all jobs are assumed to be zero and the objective is to minmize the makespan 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, in dynamic scheduling problems, the maximum completion time of all jobs is determined by the completion time of the latest-released job. Therefore, dynamic scheduling problems generally use the mean flow time of all jobs as the objective function instead of the maximum completion time of all jobs.

This paper considers two performance metrics 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 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.

3.4 Selection operartor

3.5 Improved genetic algorithm design

4 Computational Experiments

Based on the above descriptios, we validated the performance of ACO on benchmark instances. The algorithm was implemented in C++ on a Windows XP machine with Pentium IV 2.20GHz and 2G memory. Inasmuch as parameter settings have big impacts of ACO's efficiency, we identified the parameters using multiple trails and the final paramters are set as follows: the number of ants (N_ANT_COUNT) = number of tasks (N_TASK_COUNT), the maximum number of iterataions $N_{C_{max}} = 50$, $\tau_0 = 1$, $\alpha = 1.0$, $\beta = 2.0$, $\rho_1 = 0.1$, $\rho_2(t_0) = 1$, $\rho_{2_{min}} = 0.5$, $r_1 = 0.6$, $r_2 = 0.3$, $r_3 = 0.1$.

We first analyze the proposed ACO's performance based on the results for problem $C = 57$ from Kilbridge benchmark set, then present the overall performance comparison for all 24 benchmark problems in the set. The task precedence graph and computational results for problem Kilbridge are given in Figure 1 and Table 1, respectively.

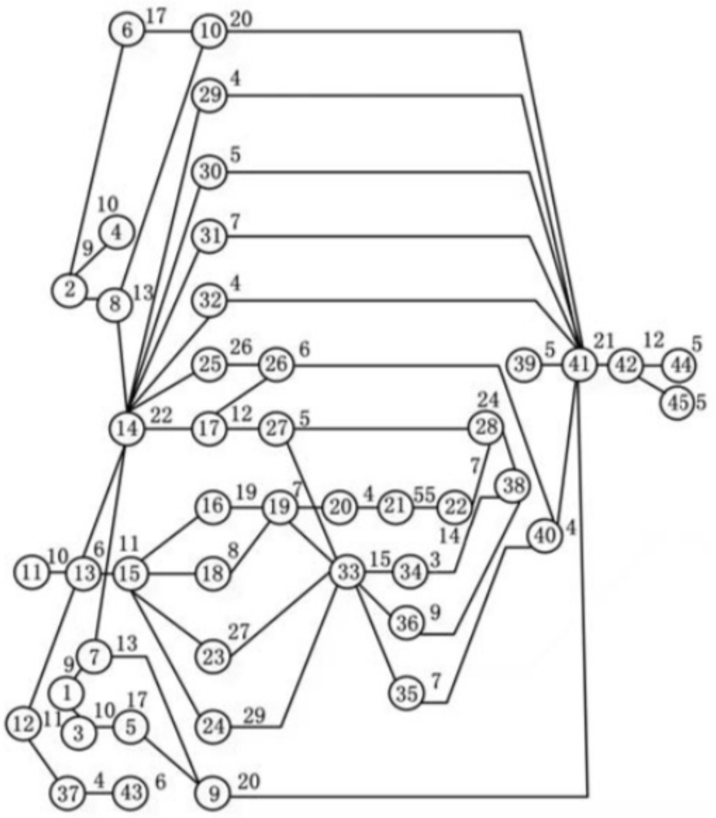

Figure 1: Task Precedence Graph of Problem Kilbridge

Table 1: Computational Result of Problem Kilbridge

Workstation No.	Tasks	Workstation Time(s)	Free Time (s)
1	1, 11, 12, 13, 15, 18, 39	55	2
2	2, 7, 8, 16	54	3
3	14, 17, 19, 20, 27, 31	57	0
4	21	55	2
5	23, 24	56	1
6	3, 4, 22, 30, 33, 34	57	0
7	5, 25, 29, 36	56	1
8	6, 26, 28, 35	54	3
9	9, 10, 32, 38, 40	55	2
10	37, 41, 42, 43, 44, 45	53	4

It can be seen from the computational results that our proposed algorithm was able to identify the solution with minimal number of workstations and balance rate of 96.8% for the problem Kilbridge. Figure 2 shows the workstation workload is well balanced and the solution was identified within 1s, which proved the efficiency of the proposed algorithm.

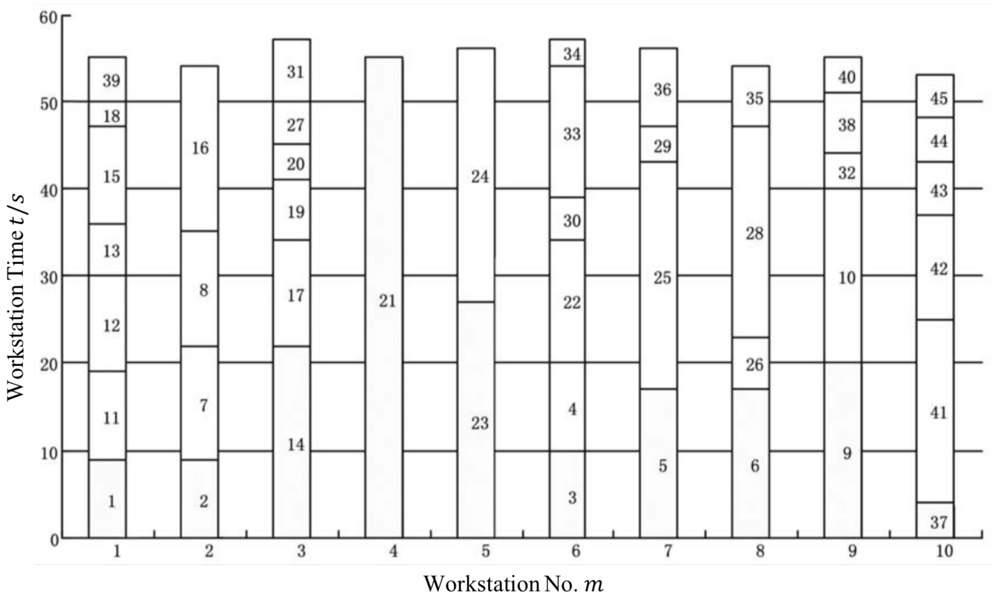

Figure 2: Task Precedence Graph of Problem Kilbridge

Table ?? shows the computational results of the proposed algorithm with weighted algorithm and genetic algorithm (GA). It can be seen from the table that both the proposed ACO and GA can identify the solution with minimal number of workstations for all the 24 benchmarking problems and their overall performance is superior than the weighted algorithm. It is worth noting that the weighted algorithm uses a static precendence rule and its selection criterion is unique and deterministic; the proposed algorithm is stochastic in nature and it can fully utilize the information between ants and previous experience, which helps the algorithm jump out of local optima and therefore find the optimal solutions.

Table 2: Computational Result Comparisons on Benchmarking Problems

Problem	No. Tasks n	Cycle C	Time t_{sum}	Min. No. Workstation m_{min}	Adaptive ACO			Weight	GA
					m	L	I		
Jackson	11	10	46	5	5	92.0%	1.095	6	5
		13		4	4	88.5%	0.707	4	4
Heskiaoff	28	205	1024	5	5	99.9%	0.447	6	5
		256		4	4	100.0%	0	5	4
Buxey	29	30	324	12	12	90.0%	2.309	12	12
		41		8	8	98.8%	0.707	9	8
		54		7	7	85.7%	4.629	7	7
Sawyer	30	41	324	8	8	98.8%	0.707	9	8
		47		7	7	98.5%	2.390	8	7
Lutzl	32	1414	14140	11	11	90.9%	134.400	12	11
		1572		10	10	89.9%	134.270	11	10
		2020		8	8	87.5%	110.640	8	8
		2828		6	6	83.3%	449.620	6	6
Kilbridge	45	57	552	10	10	96.8%	2.240	11	10
		79		7	7	99.8%	0.377	8	7
		92		6	6	100.0%	0	7	6
		110		6	6	83.6%	18.876	6	6
		138		4	4	100.0%	0	5	4
		184		3	3	100.0%	0	4	3
Tonge	70	176	3510	21	21	95.0%	8.423	22	21
		364		10	10	96.4%	4.795	11	10
		410		9	9	95.1%	6.896	10	9
		468		8	8	93.8%	15.050	8	8
		527		7	7	95.1%	14.540	7	7

5 Conclusions

This paper proposes an adaptive ACO algorithm for the assembly line balancing problem. It encompasses the following characteristics: comprehensive considerations of three rules of utilization, exploration and random search to contruct feasible assignment plans; comprehensive considerations of assembly line balancing rate as well as smoothing factor to evaluate balancing performace to increase differentiation capability of different solutions; adaptive modifications of global pheromone evaporation factor in the construction phase to help the algorithm jump out of local optima and improve its global search capability. Computationsl results validated the superior performance of the proposed algorithm.

This paper only considers single deterministic assembly balancing problem and the proposed algorithm can be used to solve more complex assembly balancing problems, like dynamic problem, stochastic problem, multi-objective problem and multi-model assembly line balancing problems.

References