2nd International Conference on Autonomous Robots and Agents
December 13-15, 2004 Palmerston North, New Zealand

An Application of Direct Coding Genetic Algorithm to Solving the Problem of Dynamic Rescheduling

Ruedee Masuchun
Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang
Ladkrabang, Bangkok, Thailand
kmrudee@kmitl.ac.th

Abstract

In a dynamic environment, it is very difficult to obtain the optimal schedule within a short time, if an optimal schedule can be obtained at all. This paper presents an application of direct coding genetic algorithm to rapidly construct a new schedule for the shop floor when the fitness functions include features that address both efficiency and stability. This is accomplished by implementing the scheduling heuristic and modified direct coding genetic algorithm resolution procedure in a simulated job shop. The results demonstrate that using direct coding genetic algorithm can provide better and faster schedule compared with other algorithm.

Keywords: dynamic rescheduling, direct coding, genetic algorithm

1 Introduction

Rescheduling is a procedure in which a new schedule is generated at various times to improve production efficiency. In a dynamic environment, it is very difficult to obtain the optimal schedule within a short time, if an optimal schedule can be obtained at all. When this is coupled with the fact that real environments are dynamic, it is easy to understand why an algorithm that can find the "good" schedule quickly is strongly preferred over all other outcomes.

When one moves from the objective of optimality to "good", things change. In the literature, the methodologies to find a good schedule can be divided into two categories: 1) match-up scheduling and 2) complete rescheduling [1]. The difference between these two approaches is that using complete rescheduling, a new schedule can be completely different from the initial schedule. However, using match-up scheduling, the part of the schedule closest to the current time is modified while that furthest from the current time is kept unchanged.

Many dynamic scheduling (or rescheduling) algorithms have been proposed that use global information to rapidly reschedule the shop floor once the rescheduling factor is valid. Most scheduling algorithms used along with the evolutionary rescheduling strategies are approximation algorithms such as tabu search (TS), simulated annealing (SA), genetic algorithm (GA). These algorithms can be utilized to obtain the near-optimal solution very quickly. Hence, in scheduling problems, these approximation algorithms can be more useful when the mathematical models cannot represent the problem domain or the problem is too large to find the timely solutions.

In this paper, the same job shop scheduling problem used in [2] is used here as an illustrative case study so that the effectiveness of the algorithms can be compared and analyzed. In [2], the fitness functions include features that address both efficiency and stability. A periodic rescheduling strategy is employed to determine when to construct a new schedule. The simulation is developed to model the problem. A direct coding genetic algorithm is implemented in a simulated job shop to obtain a good schedule for the job shop problem in a dynamic environment.

2 Job Shop Scheduling Problem

The job shop scheduling problem used in [2] can be defined as follows.

Consider a job shop with M machines, in which each machine performs a different operation. There are N jobs belonging to Q classes and each class requires from 1 to M operations in its own sequence. Each job, n, arrives at time a_n, has due date DD_n assigned according to its class, and requires $p_{n,m}$ units of processing time on the m^{th} machine of its own sequence. Jobs are executed when scheduled. That is, machines never break down. The objective is to construct schedules that effectively accommodate these dynamic arrivals by simultaneously maximizing efficiency and stability.

2.1 Schedule Heuristic

In this paper, efficiency is defined in terms of both makespan and tardiness; however, when stochastic models are used, the variances of makespan and

tardiness have been shown to be unequal [3]. They suggest that constant multipliers be introduced to alleviate this problem so efficiency becomes a linear combination of makespan and tardiness with the coefficients related to variance. This study uses their suggestion of assigning multipliers of 5 and 2 to makespan and tardiness, respectively.

Stability is also defined by two measures. The first measure is the deviation between the original job starting times and the starting time in the new schedule [4]. The second measure is a penalty function related to total deviation from the current time to reflect the fact that the cost of changing a schedule increases as changes are made closer to the current period [5].

The schedule heuristic to support a scheduling methodology can be summarized as follows.

Objective function = Max (Efficiency + Stability)

$$= \text{Min}\left\{ \begin{array}{l} 5*\left[\underset{n\in C}{\text{Max}}\left(d_n\right) - \underset{n\in C}{\text{Min}}\left(s_n\right) \right] + 2* \sum_{n\in C} \psi_n \left(d_n - DD_n\right) \\ + \sum_{m\in M} \sum_{n\in C} \left| t'_{n,m} - t_{n,m} \right| + \sum_{m\in M} \sum_{n\in C} \left(PF(\gamma'_{n,m} + t_{n,m} - 2*t) \right) \end{array} \right\} \quad (1)$$

Several requirements are needed to complete the heuristic.

1. Each job must be completely processed.

$$t'_{n,m} - t'_{j,m} \geq p_{j,m} \quad (2)$$

This requirement is enforced on each machine m for each job whose sequence includes this machine and where $t'_{n,m} > t'_{j,m}$

2. It is assumed that jobs are processed in sequence through the machines

$$t'_{n,m} - t'_{n,m-1} \geq p_{n,m-1} \quad (3)$$

This requirement is enforced on each machine m for each job whose sequence includes this machine and where m > 1.

3. Processing cannot be started until after the job arrives

$$t'_{n,m} \geq a_n \quad (4)$$

This requirement is enforced on each machine m for each job whose sequence includes this machine.

4. When more than one job arrives at the same time to a machine, the job that is first in the schedule is given a higher priority and released to the machine first. The requirement is written as follows:

If $t \geq \left(t'_{n,m} + p_{n,m} \right)$ or $t \geq a_n$,

and $t \geq \left(t'_{j,k} + p_{j,k} \right)$ or $t \geq a_j$,

and $(m+1)^{th}$ machine $= (k+1)^{th}$ machine,

and job n has the higher priority than job j

Then $t'_{n,m+1} < t'_{j,k+1}$ (5)

Given that,

a_n = Arrival time of job n
DD_n = Due-date of job n
ψ_n = 1, if $(d_n - dd_n) > 0$
0, otherwise
$t_{n,m}$ = Starting time of job n on the m^{th} machine in the original schedule π
t = Current time; time when scheduling takes place
$p_{n,m}$ = Process time of job n on the m^{th} machine
γ = Total deviation from current time
$PF(\gamma)$ = Penalty function associated with total deviation from the current time
$t'_{n,m}$ = Starting time of job n on the m^{th} machine in the new schedule π'
s_n = Starting time of job
d_n = Departure time of job

The nature and scope of the simulated job shop has been synthesized from the work of others [6, 7, 8, 9, 10, 11, 12]. The parameters required to simulate the job shop are assigned as follows.

- Number of machines is 6.
- A sequence each job will visit includes 1 to 6 machines. The number of operations for each job is assigned according to a discrete uniform distribution between 1 and 6.
- A purely random sequence is utilized to represent a more difficult control problem in a more conservative direction [11]; consequently, each job belongs to a different class and has its own sequence with its own processing time at each machine. The sequence is randomly assigned with an equal probability to initially start at any of six machines and proceed to any of the other five machines [8, 9].
- The average process time for each of the operations is assigned according to an exponential distribution [8, 9, 11, 13] with a mean of 1.0.
- The interarrival time for each job is generated from the exponential distribution with mean 0.73 [8, 9].

The tightness factor of the due-date for each job is assigned from the normal distribution with a mean of 10 and a standard deviation of 2 [8].

2.2 Simulation

A simulation model that represents this job shop scheduling problem is developed by using Visual Basic programming language. The verification and validation are performed to determine that this simulation performs as intended and is an accurate representation of the system under this study. To eliminate any biases introduced during the initial period of the simulation, the truncation point is defined so that all data of the transient phase is

discarded. In this study, the truncation point is defined as the first 230 arrivals. Therefore, each simulation run consists of 1,000 jobs with all data associated with jobs that start processing before the truncation point being discarded.

3 Direct Coding Genetic Algorithm

A genetic algorithm was first introduced to the scheduling problem by Davis [14]. It is a randomized algorithm and motivated by the natural selection and evolution from the studies of cellular automata. Results are controlled by probability and based on a random number. Genetic algorithm uses a chromosome to represent a solution. Initial populations are generated and their fitness values are calculated. Two genetic operators called crossover and mutation combine and alter current (parent) solutions to form new (child) solutions. The advantage of genetic algorithm is that it can operate on several solutions simultaneously and has fewer problems with local optima. Initializing with a good population and reasonable population size can increase the efficiency of genetic algorithm [15].

According to coding, genetic algorithm can be categorized into two approaches as follows.

- Indirect coding: a schedule can be constructed in accordance with the rule represented by each chromosome. The weakness of this approach is that the space is not entirely searched.
- Direct coding: a schedule can be constructed directly from each chromosome. The weakness of this approach is the requirement of the complex genetic operators to generate good and feasible solutions. This approach is used in this paper to find the solution. Details will be presented later.

3.1 Initialization

To start with, an initial population of Z parent schedules is randomly selected. Each schedule is defined by each chromosome represented by a matrix as shown in Figure 1. Each row corresponds to the production plan for each manufacturer. Each row represents the production schedule of a particular product during the horizon. Figure 1 displays a possible parent given that there are 4 manufacturers producing two similar products. The planning time is 2 periods.

M1	P2(1, 30)	P1(2, 5)	P2(1, 20)	P2(2, 40)
M2	P1(1, 10)	P1(2, 50)	P2(2, 25)	P2(2, 40)
M3	P2(1, 20)	P1(2, 15)	P2(1, 30)	P2(2, 5)
M4	P2(1, 25)	P1(2, 25)	P2(1, 10)	P2(2, 20)

Figure 1: A possible parent.

From figure 1, the production plan for manufacturer 1 is producing product 1 for 30 and 5 units during period 1 and period 2, respectively, and producing product 2 20 and 40 units during period 1 and period2 respectively.

3.2 Evaluation

The efficiency and stability are calculated for each parent schedule from equation (1). It is important that equations (2) to (5) must be satisfied to assure that all production plans are possible.

3.3 Selection

To generate an offspring, a pair of parent schedules is required. Therefore, to get Z offspring, Z pairs of parent schedules are randomly selected. The parent schedule having better fitness value has more chance to be selected.

3.4 Crossover

Crossover operator is applied to each pair of parent schedules to generate an offspring. In this study, a row is randomly selected on the first parent schedule and called crossover point. The production schedules belonging to those manufacturers from that row down to the bottom of the matrix are switched. Figure 2 displays the crossover between parent 1 and parent 2 when the crossover point is at row 2.

3.5 Mutation

Mutation operator is introduced to each offspring. In this study, two elements in each row are randomly selected and called mutation points. A production schedule at one position is switched with another position. Figure 3 displays the mutation for the production schedule of offspring 1 when the mutation points are 2 and 4.

The parameters required for the direct coding genetic algorithm are assigned according to [10, 14, 15]:

- Population size (Z) equals 10.
- Crossover probability is 1.0.
- Mutation probability for each string is 1.0.
- Stopping condition is satisfied after 10,000 schedules are evaluated.

4 Results

The general observation of the average results from ten simulation runs is performed first. The average result from ten simulation runs are presented in table 3 along with the results obtained from genetic local search algorithm [2]. Since direct coding genetic algorithms require the fitness function as shown in equation (1) to be maximized, the well known conversion, Min z = Max –z, was utilized. As such, numerical results reported in this section should be interpreted as the more negative a value, the better. From Table 3, it can be seen that using the direct coding genetic algorithm (GA) can obtain better

schedule compared to those obtained by using genetic local search algorithm (GLS).

Parent 1

M1	P2(1, 30)	P1(2, 5)	P2(1, 20)	P2(2, 40)
M2	P1(1, 10)	P1(2, 50)	P2(2, 25)	P2(2, 40)
M3	P2(1, 20)	P1(2, 15)	P2(1, 30)	P2(2, 5)
M4	P2(1, 25)	P1(2, 25)	P2(1, 10)	P2(2, 20)

Parent 2

M1	P2(1, 17)	P1(2, 18)	P2(1, 30)	P2(2, 30)
M2	P1(1, 30)	P1(2, 30)	P2(2, 45)	P2(2, 20)
M3	P2(1, 10)	P1(2, 25)	P2(1, 10)	P2(2, 25)
M4	P2(1, 40)	P1(2, 10)	P2(1, 5)	P2(2, 25)

Offspring 1

M1	P2(1, 30)	P1(2, 5)	P2(1, 20)	P2(2, 40)
M2	P1(1, 30)	P1(2, 30)	P2(2, 45)	P2(2, 20)
M3	P2(1, 10)	P1(2, 25)	P2(1, 10)	P2(2, 25)
M4	P2(1, 40)	P1(2, 10)	P2(1, 5)	P2(2, 25)

Offspring 2

M1	P2(1, 17)	P1(2, 18)	P2(1, 30)	P2(2, 30)
M2	P1(1, 10)	P1(2, 50)	P2(2, 25)	P2(2, 40)
M3	P2(1, 20)	P1(2, 15)	P2(1, 30)	P2(2, 5)
M4	P2(1, 25)	P1(2, 25)	P2(1, 10)	P2(2, 20)

Figure 2: Crossover operation.

Offspring 1

M1	P2(1, 30)	P1(2, 40)	P2(1, 20)	P2(2, 5)

Figure 3: Mutation operation.

Table 3: Simulation results

Experiment	Scheduling Interval Length	Objective function using GLS	Objective function using GA
1	100	-1031650	-1035235
2	200	-961133	-962356
3	300	-910926	-926458

5 Conclusion

In this paper, the same job shop scheduling problem used in [2] is used here as an illustrative case study so that the effectiveness of the algorithms can be compared and analyzed. The fitness functions include features that address both efficiency and stability. A periodic rescheduling strategy is employed to determine when to construct a new schedule. The simulation is developed to model the problem. A direct coding genetic algorithm is implemented in a simulated job shop to obtain a good schedule for the job shop problem in a dynamic environment. The results demonstrate that using direct coding genetic algorithm can provide better and faster schedule compared with other algorithm.

6 References

[1] Wu, S.D., Storer, R.H., and Chang, P.C., "A rescheduling procedure for manufacturing systems under random disruptions", *Proceedings, Joint U.S.A./German Conference on New Directions for Operations Research in Manufacturing, National Institute of Standard and Technology*, pp 292-306 (1991).

[2] Masuchun, R. and Ferrell, W.G., "Dynamic rescheduling with stability", *The Proceedings of the 5th Asian Control Conference, Melbourne 20 - 23 July 2004*, pp 1886-1894 (2004).

[3] Ishibuchi H., and Murata T., "A multi-objective Genetic Local Search algorithm and its application to flowshop scheduling", *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, Vol 28, No 3, pp 392-403 (1998).

[4] Wu S.D., Storer R.H., and Chang P.C., "One-machine rescheduling heuristics with efficiency and stability as criteria", *Computers and Operations Research*, Vol 20, No 1, pp 1-14, (1993).

[5] Lin N.P., Krajewski L., Leong G.K., and Benton W.C., "The effects of environmental factors on the design of master production scheduling systems", *Journal of Operations Management*, Vol 11, pp 367-384, (1994).

[6] Baker K.R., "Sequencing rules and due-date assignment in a job shop", *Management Science*, Vol 30, No 9, pp 1093-1104, (1984).

[7] Baker K.R., and Bertrand J.W.M., "A comparison of due-date assignment rules", *IIE Transactions*, Vol 13, pp 123-130, (1981).

[8] Blocher J.D., Chhajed D., and Leung M., "Customer order scheduling in a general job shop environment", *Decision Science*, Vol 29, No 4, pp 951-977, (1998).

[9] Chang F.R., "A study of due-date assignment rules with constrained tightness in a dynamic job shop", *Computers & Industrial Engineering*, Vol 31(1/2), pp 205-208, (1996).

[10] Morton T.E., and Pentico D.W., *Heuristics Scheduling Systems*, John Wiley & Sons, New York, (1993).

[11] Ragatz G.L., and Mabert V.A., "An evaluation of order release mechanisms in a job-shop environment", *Decision Sciences*, Vol 19, No 1, pp 167-189, (1988).

[12] Rohleder T.R., and Scudder G.A., "Comparison of order-release and dispatch rule for the dynamic weighted early/tardy problem", *Production and Operations Management*, Vol 2, No 3, pp 221-238, (1993).

[13] Pegden C.D., Shannon R.E., and Sadowski R.P., *Introduction to Simulation Using SIMAN*, McGraw-Hill, New York, (1995).

[14] Davis, L., "Job shop scheduling with genetic algorithms", *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pp 136-140, (1985).

[15] Chen, C.L., Vempati, V.S., and Aljaber, N., "An application of genetic algorithms for flow shop problems", *European Journal of Operational Research*, Vol 80, pp 389-396, (1995).