Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005

RESEARCH ON 3MO-BASED GENETIC ALGORITHM FOR SOLVING ORDER PLANNING

TAO ZHANG [1], YUE-JIE ZHANG [2], MENG-GUANG WANG[3]

[1] School of Information Management & Engineering, Shanghai University of Finance & Economics, Shanghai 200433, China
[2] School of Computer Science & Engineering, Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, Shanghai 200433, China
[3] School of Information Science and Engineering, Northeastern University, Shenyang 110006, China
E-MAIL: taozhang@mail.shufe.edu.cn, yjzhang@fudan.edu.cn

Abstract:

This paper proposes a mixed integer-programming model for order planning of iron-steel enterprise. Because this problem is a kind NP-hard problem and the size of this problem is bigger, the genetic algorithm based on repeatable natural scale code and 3-mutation operator is presented to solve the model. For ensuring the quality of copy strategy of the genetic algorithm, this paper develops an improving selection function for copying. Finally, this paper uses the order planning of Shanghai Baoshan iron-steel enterprise as an example to test the model and the algorithm. The numerical analysis shows that the model comes up to the production process, the solutions obtained by this algorithm are superior to those obtained by the human-machine system. So, the model and the algorithm are valid.

Keywords:

Order planning (OP); mixed integer programming; genetic algorithm (GA); 3-mutation operator (3MO); production process

1. Introduction

The production planning system of steel plants includes year-planning, month-planning, order planning and day scheduling. The order planning arranges production of orders according to the requirement of customers and capacity of machines with the result of year-planning, then day-scheduling schedules the production concretely according to the results of the order planning and considering the technical rules of the production. The main task of order planning is to plan the production period of an order according to the delivery of the order and the capacity of every machine. When the route of an order has been determined, the order planning will determine the production time of the order on every working procedure.

Many researchers have done much research about programming the order planning. C.N.Redwine applied the mixed integer-programming model and Bender Partitioning algorithm to find minimum total tardiness for order planning [1]. They introduced into the tardiness penalty list. The penalty list can show the important degree of the orders. Teng Xue analyzed the production process of the hot milling set of the steel tube plant [2]. He took the month as the minimum time unit and developed the order assignment model and carried on the optimization analysis and test.

This paper presents a mathematical model and algorithm of the order planning. The research work has character as follows: (1) Taking five-day as the smallest time unit, minimizing the total earliness (i.e., ahead of due date) penalty and tardiness penalty of the orders, an mixed integer programming model of the order planning is presented. (2) The model is solved using the genetic algorithm based on the repeatable natural number code and the 3-mutation operator. (3) The model and algorithm are tested through experiments using practical data. The result analysis shows the model and algorithm are valid.

2. Mathematical Model of Order Planning

2.1. Description of the Order Planning

The order planning can be described as follows mathematically. Suppose that there are N orders and M working procedures for the steel plant production. The weight, due date and production route (i.e., the working procedures and the optional machines of every working procedure) of every order is known. The aggregated working procedures passed by every order are the same, but the machine passed by each order on one working procedure may be different. The capacity of every machine is known. Each order must pass all procedures. Each order can be processed by one and only one machine in one working procedure. Each machine can simultaneously process only one order at most. The OP will decide the machine passed by each order in every procedure and the time each order passes every working procedure while satisfying capacity constraint and precedence relation between procedures, so that to minimize the total earliness penalty and tardiness penalty of the orders.

From the point of view of planning and scheduling theory: If the machine of the working procedure passed by every order is determinate, this is a general Flow Shop (FS) problem. If there is more than one machine that can be chosen by one order in one working procedure this is a Flexible Flow Shop (FFS) problem. The FS problem is relatively simpler. The FFS problem is much more complicated than the FS problem. The FFS problem is an extension of the traditional FS problem and exists generally in the steel-iron industry and petrochemical industry. It is first defined by Salvador based on findings from petrochemical industry [3]. In this paper we treat the order planning as FFS problem. The order planning plans the time segment (i.e., unit) each order is processed, but does not arrange the sequence of the orders in each time segment.

2.2. Mathematical Model

The mathematical model is presented on the assumptions as follows:

(a) Take the five-days as the time unit for the due date and processing time.

(b) The total usable machine-hour of each machine set of each procedure in every time segment (five-days) is known.

(c) For different machine in the same working procedure, the expended capacity for the same order is different.

(d) At each working procedure, the processing time of each order is less than five days. Following the production route, several working procedures of one order can be processed in the same time segment (five-days).

(e) The middle stock isn't considered, because it has been considered into every working procedure.

Notations:

N denotes total quantity of the orders.

M denotes quantity of procedures.

M_j denotes quantity of parallel machine in working procedure j .

T denotes planning period.

$[d_i - u_i, d_i + v_i]$ denotes delivery due date window of order i (known), where d_i is due date.

t_{ij} denotes the processing time of the order i on the working procedure j .

e_i denotes the actual weight of the order i (known).

e_{ijk} denotes the expended capacity of the order i on the machine k of the working procedure j (known).

c_i denotes the completion time of order i , i.e., the time segment that the order i passes the last working procedure.

E_{jkt} denotes the total usable machine-hour of the machine k of the working procedure j in the time segment t (known).

$$x_{ijkt} = \begin{cases} 1, & \textit{order i is processed on the machine k of} \\ & \textit{the procedure j in the time segment t} \\ 0, & \textit{otherwize} \end{cases}$$

Where, x_{ijkt} is decision variable.

i denotes the order quantity, j denotes the working procedure quantity, k denotes the machine quantity, t denotes time segment.

$$i = 1,\dots,N \qquad j = 1,\dots,M$$
$$k = 1,\dots,M_j \qquad t = 1,\dots,T$$

The mathematical model is stated as follows:

$$\min \sum_{i=1}^{N} e_i(\alpha_i \max\{0, d_i - u_i - c_i\} + \beta_i \max\{0, c_i - d_i - v_i\}) \quad (1)$$

s.t.

$$\sum_{t=1}^{T} \sum_{k=1}^{M_j} x_{ijkt} = 1 \quad (2)$$

$$\sum_{i=1}^{N} (x_{ijkt} * e_{ijk}) \le E_{jkt} \quad (3)$$

$$c_i = t_{iM} \quad (4)$$

$$t_{ij} = \{t \mid \sum_{k=1}^{M_j} x_{ijkt} = 1, \quad t = 1,2,\dots,T\} \quad (5)$$

$$t_{i,j-1} \le t_{ij}, \qquad j = 2,\dots,M \quad (6)$$

$$x_{ijkt} \in \{0,1\}, \quad (7)$$

Where, α_i denotes the earliness penalty coefficient of the order i , and β_i denotes the tardiness penalty coefficient of the order i . The goal is to minimize the total earliness penalty and tardiness penalty of all orders. The constraint (2) ensures that each order must pass every working procedure and each order must and can only be processed by one machine in each working procedure. The constraint (3) ensures that the total quantity of the orders processed by one machine cannot exceed the capacity of this machine in every five-day. The equation (4) shows what c_i is. The equation (5) shows that the time when the order i is processed in working procedure j. The constraint (6) ensures that the time when the order i is processed in working procedure j does not earlier than the time when this order is processed in working procedure $j-1$. The formula (7) denotes the domain of the decision variables.

3. Design of the Algorithm

Our order-planning model is a non-linear mixed integer-programming model. For the variable dimension is so many and the goal function is non-linear, the calculation complexity is increased. There are hundreds of orders requiring to be planned in our problem, so it is difficult to be solved by the accurate algorithm. It had been proved that FFS problem is a kind of NP-hard problem even if there are two working procedures in this problem and there are parallel machines on one of them and the goal function is to minimize makespan [4]. FFS problem is difficult to be solved to get the optimal solution by the accurate algorithm in a reasonable time [5]. The genetic algorithm has been used for the combination optimum problems, e.g., TSP and flow-shop, etc [6]. It can obtain the global optimum search using the genetic operations, e.g., selection and crossover and mutation etc. The genetic algorithm doesn't decrease the search space, but it can search bigger space in shorter time for the parallel colony search. So, the genetic algorithm will be used to solve the order-planning model in this paper.

3.1. Encoding of Chromosome

In this paper, a type of the repeatable natural number code is used to design the gene a_{ijk} of the chromosome.

$A=\{a_{ijk}\}$,

Where, $i=1,...,N$, $j=1,...,M$, $k=1,...,M_j$.

a_{ijk} denotes the time segment when the order i passes the machine k in the working procedure j.

The chromosome A can be expanded as follows:

$\{a_{111},a_{112},...,a_{11M_1},...,a_{1M1},...,a_{1MM_M},...,a_{NMM_M}\}$

Where, we call the $\{a_{i11},a_{i12},...,a_{i1M_1},...,a_{iM1},a_{iM2},...,a_{iMM_M}\}$ as a big segment, and call the $\{a_{ij1},a_{ij2},...,a_{ijM_j}\}$ as a small segment. There is only one nonzero gene in every small segment. (If the gene is zero, this machine does not process the corresponding order.) This kind of code ensures the constraint (2) is satisfied. The nonzero gene is a time value (the time when the order i passes the machine k of the procedure j). In one big segment, the scale of the value of every nonzero gene is a natural number which scale from 1 to ($T+t_d$) and the values between every two small segments must satisfy the constraint (4). In different big segments, the value can be repeatable. The completion time C_i is the value of the nonzero gene of the M th small segment $\{a_{iM1},a_{iM2},...,a_{iMM_M}\}$ of the i th big segment.

3.2. Initial population

According to the demand of the chromosome code, the initial population P(t), that is consist of L chromosomes, is produced by stochastic method.

First, produce a stochastic integer k_{11} whose scale is from 1 to M_1, and take k_{11} as the number of the nonzero element. Then, a stochastic integer whose scale is from 1 to $T+t_d$ is produced as the value of the element $a_{11k_{11}}$. In this way, the first small segment is produced. Then a stochastic integer k_{12} whose scale is from 1 to M_2 is produced as the number of the element that is nonzero. Again, a stochastic integer whose scale is from $a_{11k_{11}}$ to $T+t_d$ is produced as the value of the element $a_{12k_{12}}$. In this way, the second small segment is produced. In the same way, other small segments of the big segment $\{a_{111},a_{112},...,a_{11M_1},...,a_{1M1},a_{1M2},...,a_{1MM_M}\}$ are produced. (Here, t_d is an assistant integer.)

Repeat (1) till other big segments of this chromosome are produced. In this way, a chromosome is produced.

Repeat (1) and (2) till other L-1 chromosomes are produced.

3.3. Copy strategy

The goal function of the l th chromosome of the population is:

$$F_l=\sum_{i=1}^{N}e_i(\alpha_i\max\{0,d_i-u_i-c_i\}+\beta_i\max\{0,c_i-d_i-v_i\}) \tag{8}$$

For limiting the quantity of the chromosomes that do not satisfy the constraint (3) in the new population, we design a special selection function for copying. Define that the infeasible distance ID_l of each chromosome about the constraint (3) denotes the unfeasibility degree of the chromosome l.

$$ID_l = \sum_{j=1}^{M}\sum_{k=1}^{M_j}\sum_{t=1}^{T}\max\{(\sum_{i=1}^{N} X_{ijkt}e_{ijk} - E_{jkt}),0\} \tag{9}$$

Then, the selection function f_l for copying is:

$$f_l = C - F_l - \gamma ID_l \tag{10}$$

Where

$$C = \max_{l\in Q}\{F_l + \gamma\sum_{j=1}^{M}\sum_{k=1}^{M_j}\sum_{t=1}^{T}\max\{(\sum_{i=1}^{N} X_{ijkt}e_{ijk} - E_{jkt}),0\}\} + \lambda \tag{11}$$

$Q = \{1,2,...,L\}$ denotes the set of the chromosome in the population.

L denotes population size.

γ denotes unfeasible penalty coefficient about the constraint (3).

λ denotes an appropriate positive integer.

In this algorithm, the roulette method is used for copying. The selection probability for copying can be obtained according to the formula (12) as follows.

$$P_l = \frac{f_l}{\sum_{a=1}^{L} f_a} \qquad l = 1,2,...,L \tag{12}$$

If the value of C is only taken as a bigger constant, the value of f_l obtained will cover up the quality difference among the chromosomes when the goal values of the chromosomes are not big and there is greater difference among them. So, the value of C is obtained by the formula (11) in the course of copying.

3.4. Crossover operator

According to the crossover probability P_c, the Part Segment Crossover（PSC） is adopted for crossover. The big segments between two stochastic positions of two chromosomes are exchanged. (Crossover only exchanges the values that are nonzero of the corresponding small segments, but does not change the positions of the nonzero values.)

3.5. Mutation operator

Because there are big segments and small segments in every chromosome, the mutation course cannot be completed by a common mutation operator. For ensuring the global search and satisfying the constraint (2) and (4), the 3-mutation operator (include mutation among the big segments, mutation in the small segments, mutation in the big segments) is produced in this algorithm. The 3-mutation operator changes the positions and values of the nonzero genes of the chromosomes, where the mutation 2 achieves the position change while the mutation 1 and 3 achieve value change. At the same time, the course of selecting better chromosome of the new chromosomes obtained by mutation is completed according to the fitness function as follows.

$$f_l' = C' - F_l - \gamma\sum_{j=1}^{M}\sum_{k=1}^{M_j}\sum_{t=1}^{T}\max\{(\sum X_{ijkt}e_{ijk} - E_{jkt}),0\} \tag{13}$$

Where, C' is an adequately big constant.

Mutation 1: mutation among the big segments

Taking the big segment as the unit, the 2-exchange mutation is accomplished among the big segments B_i of the chromosome $A' = \{B_1, B_2,..., B_N\}$ according to the mutation probability P_{m1}.

(a) For the first chromosome of the population, suppose $a = 1$.

(b) A stochastic decimal fraction whose scale is from 0 to 1 is produced randomly. If this decimal fraction isn't bigger than P_{m1}, then the chromosome will be mutated, turn to (c). Otherwise, turn to (d).

(c) Every two big segments are exchanged respectively (The corresponding values that are nonzero are exchanged only, but the positions of the values are not exchanged.) and the best exchange that gets a new chromosome with the bigger fitness is selected. If the fitness of the new chromosome obtained by this best exchange is bigger than the fitness of the original chromosome, the original chromosome is replaced with the new chromosome. Otherwise, the new chromosome is not accepted. Then, the mutation in the small segments (mutation 2) and the mutation in the big segments (mutation 3) begin to proceed. Then turn to (d).

(d) If $a < L$, then $a = a + 1$, turn to (b). Otherwise, end.

Mutation 2: mutation in the small segments

Suppose chromosome $A' = \{B_1, B_2,..., B_N\} = A$.

Where, $B_i = \{a_{i11},..., a_{i1M_1},..., a_{iM1}, a_{iM2},..., a_{iMM_M}\}$ is the i th big segment.

Suppose $B_i = \{S_{i1}, S_{i2},..., S_{iM}\}$ denotes the big segment, $S_{ij} = \{a_{ij1}, a_{ij2}, ..., a_{ijM_j}\}$ denotes the small segment. According to the mutation probability P_{m2}, the 2-exchange mutation is executed among the genes of every S_{ij} of the chromosome. The detailed process is illustrated as follows:

(a) To this chromosome, suppose the small segment is $j = 1$.

(b) A decimal fraction whose scale is from 0 to 1 is produced randomly. If this decimal fraction isn't bigger than P_{m2}, the chromosome will be mutated, turn to (c). Otherwise, turn to (d).

(c) The nonzero values are put on other $M_j - 1$ positions respectively, and the best exchange is selected. If the fitness of the new chromosome obtained by the best exchange is bigger than the fitness of the original chromosome, the original chromosome is replaced with the new chromosome. Otherwise, the new chromosome is not accepted. Then, turn to (d).

(d) If $j < M$, then $j = j + 1$, turn to (b). Otherwise, end.

Mutation 3: mutation in the big segments

(a) To this chromosome, suppose the big segment is $i = 1$.

(b) A decimal fraction whose scale is from 0 to 1 is produced randomly. If this decimal fraction isn't bigger than P_{m3}, the chromosome will be mutated, turn to (c). Otherwise, turn to (d).

(c) First, an integer whose scale is from 1 to $T + t_d$ is produced randomly again as the value of the element $a_{i1k_{i1}}$. Then, an integer whose scale is from $a_{i1k_{i1}}$ to $T + t_d$ is produced randomly as the value of the element $a_{i2k_{i2}}$. In the same way, the other small segments of the big segment $\{a_{111}, a_{112}, ..., a_{11M_1}, ..., a_{1M1}, a_{1M2}, ..., a_{1MM_M}\}$ are produced. If the fitness of the new chromosome obtained by this way is bigger than the fitness of the original chromosome, the original chromosome is replaced with the new chromosome. Otherwise, the new chromosome is not accepted. Then, turn to (d).

(d) If $i < N$, then $i = i + 1$, turn to (b). Otherwise, end.

3.6. Character and Procedure of the Algorithm

In conclusion, this algorithm improved the traditional genetic algorithm in the following aspects:

(a) A type of the repeatable natural scale code is produced, and the constraint (2) and (4) are satisfied.

(b) The 3-mutation operator proposed in this algorithm can complete the whole mutation process effectively.

(c) The whole process of this algorithm ensures the feasibility of the constraint (2) and (4).

(d) For satisfying the computational demand, the function for copying and the fitness function for evaluating are different. To the different population, the value C is a variable with formula (8). No matter what the chromosomes change, the formula (8) can represent the different quality among the chromosomes. So, it ensures effective copying function.

(e) In the formula (13), which denotes the fitness function, the objective function of the model and the infeasible distance about the constraint (3) are considered. Selecting better solution is to select the solution with smaller objective function value and smaller infeasible distance, but doesn't refuse the solution whose infeasible distance is nonzero to enter the new population. This search method includes the interior-point method (IPM) and the outer-point method (OPM). The IPM searches from the infeasible solution to the feasible solution while the OPM searches from the feasible solution with worse quality to the feasible solution with better quality. This hybrid search method ensures the global property of searching.

4. Experimental Results

This paper takes the order planning of ShangHai Baoshan iron-steel enterprise as an example to test our algorithm.

In the practical production, the planner will plan the orders that must be finished in two months, so T =12. This paper uses the examples with N = 51, N = 138 and $N = 249$ respectively to test our model and algorithm. The algorithm is programmed in C++ and runs on a personal computer, where $M = 3$, $M_1 = M_2 = 1$, $M_3 = 5$, the biggest generation MAXGEN is 100, the population size is 20, $P_c = 0.3$, $P_{m1} = 0.3$, $P_{m2} = 0.3$, $P_{m3} = 0.3$, and the infeasibility penalty coefficient $\gamma = 10$. Table 1 shows the computational results.

Table 1. Experimental results

n	*VMM*	*VGA*			E_n	T_n	*t*
		Bf	*Wf*	*Mf*			min
51	12800	3475	4100	3725	3	0	3
138	32500	10800	12350	11300	8	2	8
249	100800	31470	36210	33570	19	5	26

Where, n denotes order quantity, VMM denotes the goal value obtained by man–machine method, Bf denotes the best value, Wf denotes the worst value, Mf denotes the average value, VGA denotes the goal value obtained by the genetic algorithm, E_n denotes quantity of the earliness orders in Bf, T_n denotes quantity of the tardiness orders in Bf, t denotes computation time.

From Table 1, the best solution obtained by our algorithm is superior to the solution obtained by the man-machine method. The earliness and tardiness order quantity is smaller. For example, the test result of 51 orders shows that the earliness order quantity is 3 and the earliness time is all five days, the tardiness order quantity is 0.

For showing the variety of the goal value and the unfeasibility penalty, we take the 51 orders as the example. Figure 1 shows the graph of the objective (F) and the unfeasibility penalty (B) of the best solution. When B=0, the solution is feasible solution.

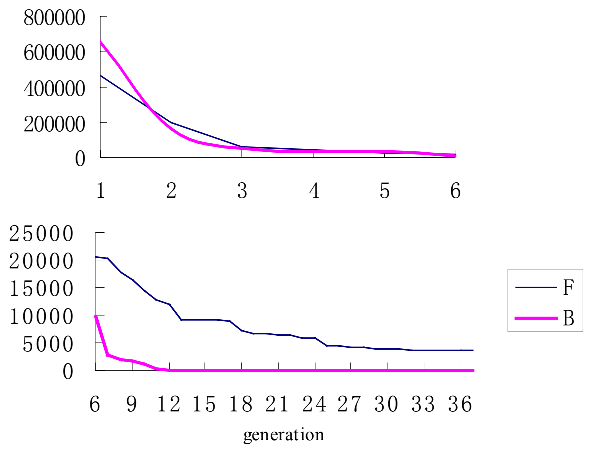

Figure 1. Graph of the objective (F) and the unfeasibility penalty (B) (γ=10)

Figure 1 shows that the convergent speed of B is very high and B reaches 0 at the 13th generation, i.e. the feasible solution is obtained at the 13th generation. The convergent speed of F is lower than the convergent speed of B, F is near to the best value at the 25th generation, and F reaches the best at the 34th generation and doesn't change clearly any longer.

5. Conclusion

Taking the five-day as the smallest time unit, minimizing the total earliness penalty and tardiness penalty of the orders as the target, we develop a mixed integer programming mathematical model of the order planning.

The genetic algorithm based on repeatable natural scale code and 3-mutation operator is presented. Adopting the search method including IPM and OPM, we solve the problem successfully.

The computational results show that: the best solution obtained by the algorithm is superior to the solution obtained by the man-machine method; the relative difference between the worst value and the best value does not exceed 20%; the relative difference between the average value and the best value does not exceed 8%; the earliness and tardiness order quantity is very small and the computational speed is very fast. Thus, our algorithm is proven to be effective.

Through the experimental analysis, this paper recommends the infeasible penalty coefficient $\gamma = 8 \sim 15$.

Acknowledgements

This paper is supported by National Natural Science Foundation of China (No.60203010) and “211 Project” of Shanghai University of Finance & Economics.

References

[1] C. N. Redwine, and D. A. Wismer, “A mixed integer programming model for scheduling orders in a steel mill”, Journal of Optimization Theory and Applications, Vol 14, No. 3, pp. 305-318, 1974.

[2] X. Teng, “The production structure model of the hot-milling set, the order distribution model and the optimum analysis of the steel tube plant”, System Engineering theory and practice (Chinese), No. 2, pp. 40-48, 1985.

[3] M. S. Salvador, “A solution to a special class of flow shop scheduling problem”, Proceedings of Symposium of the Theory of Scheduling and Applications, Springer-Verlag, Berlin, pp. 83-91, 1973.

[4] J. Hunsucker, and J. Shah, “Performance of priority rules in a due date flow shop”, OMEGA, Int. J. of Mgmt. Sci., Vol. 20, No. 1, pp. 73-89, 1992.

[5] S. C. Chang, “Scheduling flexible flow shops with no set-up effects”. IEEE Transactions on Robotics and Automation, Vol. 10, No. 2, pp. 112-122, 1994.

[6] I. Lee, R. Sikora, and J. S. Michael, “A genetic algorithm-based approach to flexible flow-line scheduling with variable lot sizes”. IEEE Transactions on Systems, Man, and Cybernetics — Part B: Cybernetics, Vol. 27, No. 1, pp. 36-54, 1997.