The Fifteenth International Symposium on Artificial Life and Robotics 2010 (AROB 15th '10),
B-Con Plaza, Beppu,Oita, Japan, February 4-6, 2010

Production Scheduling Based on Mixed-Integer Evolutionary Algorithm

Yung-Chin Lin[a,b], Yung-Chien Lin[a], Kuo-Lan Su[b]

[a]*Department of Electrical Engineering, WuFeng Institute of Technology, Chiayi, Taiwan*
[b]*Department of Electrical Engineering, National Yunlin University of Science & Technology, Yunlin, Taiwan*
e-mail: {yclin@mail.wfc.edu.tw; chien-lin@mail.wfc.edu.tw; sukl@yuntech.edu.tw}

Abstract: Production scheduling is one of the most important decision-making problems in the manufacturing industry. The problem is complex due to coupling with combinatorial property and constrained requirements. To describe production scheduling, a mixed-integer nonlinear programming (MINLP) model is developed to formulate this decision-making problem. On the other hand, in order to effectively make an optimal decision, a mixed-integer evolutionary algorithm is proposed to solve this MINLP problem. Finally, an experimental example is used to test the algorithm. The experimental results demonstrate the proposed algorithm can effectively handle the production scheduling problem.

Keywords: production scheduling, mixed-integer nonlinear programming, evolutionary algorithm**.**

I. Introduction

In a batch manufacturing system, production scheduling is one of the most important decision-making problems. In such a manufacturing system, batch operations (e.g. batch drying, batch distillation, batch reactors, etc.) are often used to produce multiple products that are similar in nature. When products are similar in nature, they need identical processing steps and the same series of processing units. Therefore, in order to maximize productivity in the shortest possible time, the optimal process scheduling is crucial in such a sequential processing decision problem.

Due to the coupling with combinatorial property and conflict constraints, this decision-making problems is complex and difficult to solve. In order to describe this decision-making problem, we develop a mixed-integer nonlinear programming (MINLP) model, based on Karimi's models [1], to formulate this production scheduling problem. In the MINLP model, continuous variables are used to describe the interactive relationships (e.g. mass balances, energy balances, physical phenomena, etc.), and integer variables are used to represent the existence of processes and the operational status of the processing units. Owing to the combinatorial property of production-strategy selection together with conflict constraints, the MINLP model presents complex characteristics such as multimodality, large dimensionality, strong nonlinearity, nonconvexity, and nondifferentiability. As a result, it is difficult to find the globally optimal solution.

Evolutionary algorithms (EAs) [2], [3] are defined as a class of stochastic search and optimization methods that begin a population of randomly generated solutions and evolve towards an optimal solution by repeatedly applying a set of genetic operations. Due to their ability to escape from the local optimal solutions, EAs have been demonstrated a promising candidate for solving complex optimization problems, including the constrained optimization problems. For optimization problems with complicated constraints, Michalewicz and Schoenauer [4] surveyed and compared several constraint-handling techniques used in EAs. Of these techniques, the penalty function method is one of the most popularly used techniques to handle the constraints. In this method, the fitness function including a penalty function, i.e. the squared or absolute constraint violation term, is used to reject infeasible solutions. However, these penalty function methods have a fatal weakness when the penalty parameters are large. For example, if the penalty parameters are large, the penalty function tends to be ill-conditioned near the boundary of feasible domain. Thus it may lead to a local solution or an infeasible solution. In this paper, to effectively solve mixed-integer constrained optimization problems (or MINLP problems), a mixed-integer evolutionary algorithm based on Lagrange method is developed for solving the MINLP problems as production scheduling problems.

In this paper, we not only formulate two MINLP models to describe the production scheduling problems, but also propose a mixed-integer evolutionary algorithm to handle them. The proposed algorithm has been successfully applied to a number of mixed-integer optimization problems [5], [6]. Finally, a production scheduling problem presented by Karimi [1] is employed to test the performance of the proposed method. The computational results demonstrate that the proposed method performs much better than the penalty method.

II. MINLP Formulation for Production Scheduling

Karimi [1] proposed a mixed-integer linear programming (MILP) model to formulate the production scheduling problem. In the following, we will describe this problem for details, and then propose a complete and explicit MINLP model to handle the production scheduling problem.

In this paper, we investigate a problem of scheduling *N* products across an *M*-stage serial processing plant with a single unit per stage. The configuration of this multiproduct batch plant is shown in Figure 1. In this batch plant, no storage (buffer) is available between the processing units. Therefore, if a product finishes processing on a unit and the downstream unit is not free (i.e. still processing a previous product), then the completed product must be held in the unit until the downstream unit becomes free. In this multiproduct plant, all products pass through the series of *M* units and the processing conditions are known *a priori* for all the products (i.e., the processing time t_{kj} of the product P_k on unit *j* is specified). Based on these specified processing conditions, the objective of the scheduling problem is to obtain the order in which the batches should be produced so as to maximize the productivity of the process, i.e., to minimize the makespan (the total time required to produce all the batches).

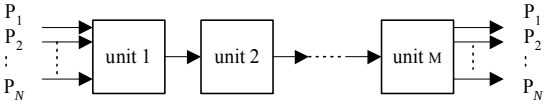

Fig.1. Configuration of batch plants.

Firstly, let C_{ij} denote the time at which the *i*th product in the production sequence leaves unit *j*. Hence, C_{NM} is the makespan of the given product sequence. To compute C_{ij}, consider the time-space relation of neighborhood stages and successive products. The completion time of the *i*th product in the sequence on unit *j* is simply the time at which unit *j* starts processing the *i*th product plus its processing time p_{ij}. But unit *j* cannot start processing the *i*th product until it has processed the previously (*i*-1)th product, or until the *i*th product has been processed by the upstream unit (*j*-1), as shown in Figure 2. We therefore have the following recurrence relations except for unit 1:

$$C_{ij} = \max\left[C_{(i-1)j}, C_{i(j-1)}\right] + p_{ij} \quad \text{for } i = 1,\ldots,N;\ j = 2,\ldots,M \tag{1}$$

For unit 1, the completion times for each product can be obtained as follows:

$$C_{i1} = \max\left[C_{(i-1)1}, C_{(i-2)2}\right] + p_{i1}, \quad \text{for } i = 1,\ldots,N \tag{2}$$

Note that $C_{ij} = 0$ for $i \le 0$ or $j \le 0$.

To order the production sequence of products, a set of binary variables are introduced as follows:

$$Y_{ki} = \begin{cases} 1, & \text{if product } \mathrm{P}_k \text{ is at position } i \\ & \text{in the production sequence} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

for $k = 1,\ldots,N;\ i = 1,\ldots,N$

For example, $Y_{32} = 1$ means that product P_3 is the second in the production sequence, and $Y_{32} = 0$ means that it is not in the second position. Based on the significance of Y_{ki}, we obtain the following two sets of equality constraints.

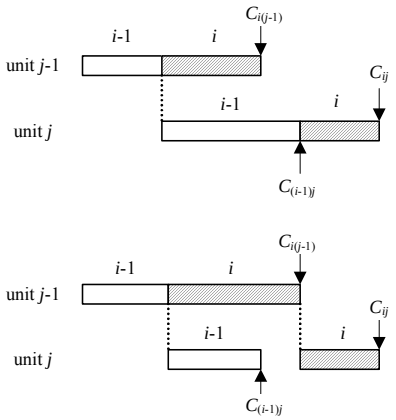

Figure 2. The time-space relation of C_{ij}, $C_{i(j-1)}$ and $C_{(i-1)j}$.

The first set of constraints ensures that a product is assigned to one and only one position in the processing sequence:

$$\sum_{i=1}^{N} Y_{ki} = 1 \quad k = 1,\ldots,N \tag{4}$$

The second set of constraints ensures that a position in the sequence is assigned to one and only one product:

$$\sum_{k=1}^{N} Y_{ki} = 1, \quad i = 1,\ldots,N \tag{5}$$

Let t_{kj} denote the processing time for product P_k on unit *j*. Now we must utilize a given set of Y_{ki} and t_{kj} to determine the processing time p_{ij} of the *i*th product in the production sequence on unit *j*. If product P_k in the sequence position *i*, then p_{ij} must be t_{kj}. On the other hand, we know that if product P_k in the sequence position *i*, then $Y_{ki} = 1$ and $Y_{k1} = Y_{k2} = \cdots = Y_{k(i-1)} = Y_{k(i+1)} = \cdots = Y_{kN} = 0$. Therefore, the processing time p_{ij} can be represented as:

$$p_{ij} = Y_{1i}t_{1j} + Y_{2i}t_{2j} + \cdots + Y_{Ni}t_{Nj} \quad \text{for } i = 1,\ldots,N;\ j = 1,\ldots,M \tag{6}$$

Replacing (1) and (2) by (6), so we have:

$$C_{ij} = \max\left[C_{(i-1)j}, C_{i(j-1)}\right] + \sum_{k=1}^{N} Y_{ki}t_{kj} \quad \text{for } i = 1,\ldots,N;\ j = 2,\ldots,M \tag{7}$$

$$C_{i1} = \max\left[C_{(i-1)1}, C_{(i-2)2}\right] + \sum_{k=1}^{N} Y_{ki}t_{k1} \quad \text{for } i = 1,\ldots,N \tag{8}$$

Since the criterion is to minimize the makespan, the scheduling problem can be formulated as a MINLP problem as follows:

$$\min C_{NM} \quad \text{subject to Eqs. (4), (5), (7) and (8)} \tag{9}$$

III. Mixed-Integer Evolutionary Algorithm

Let us consider a mixed-integer nonlinear constrained optimization problem as follows:

$$\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x},\mathbf{y}) \tag{10}$$

subject to

$$h_j(\mathbf{x},\mathbf{y}) = 0,\ j = 1,...,m_e \tag{11}$$

$$g_j(\mathbf{x},\mathbf{y}) \le 0,\ j = 1,...,m_i \tag{12}$$

where $\mathbf{x}$ represents an n_C-dimensional vector of continuous variables, $\mathbf{y}$ is a n_I-dimensional vector of integer variables, and $h_j(\mathbf{x},\mathbf{y})$ and $g_j(\mathbf{x},\mathbf{y})$ stand for the equality and inequality constraints. To abbreviate these expressions, a compact notation $\mathbf{z} = (\mathbf{x},\mathbf{y})$ is used in the following discussions, and the problem is referred to as primal problem.

The Lagrange function corresponding to the primal problem is defined by

$$L(\mathbf{z},\lambda,\mu) = f(\mathbf{z}) + \sum_{k=1}^{m_e} \lambda_k h_k(\mathbf{z}) + \sum_{k=1}^{m_i} \mu_k g_k(\mathbf{z}) \tag{13}$$

where λ_k and μ_k are the Lagrange multipliers. The exterior penalty term can be used to define the new objective function, termed the augmented Lagrange function, as

$$L_a(\mathbf{z},\nu,\upsilon) = f(\mathbf{z}) + \sum_{k=1}^{m_e} \alpha_k \left\{ \left[h_k(\mathbf{z}) + \nu_k \right]^2 - \nu_k^2 \right\} + \sum_{k=1}^{m_i} \beta_k \left\{ \left\langle g_k(\mathbf{z}) + \upsilon_k \right\rangle_+^2 - \upsilon_k^2 \right\} \tag{14}$$

where α_k and β_k are positive penalty parameters, the bracket operation is denoted as $\langle g \rangle_+ = \max\{g, 0\}$, and the corresponding Lagrange multipliers $\nu = (\nu_1,...,\nu_{m_e})$ and $\upsilon = (\upsilon_1,...,\upsilon_{m_i}) \ge \mathbf{0}$ are defined as $\lambda_k = 2\alpha_k \nu_k$ and $\mu_k = 2\beta_k \upsilon_k$.

In nonlinear programming, the Kuhn-Tucker optimality conditions are used to solve constrained optimization problems. The Kuhn-Tucker optimality conditions are suitable only for differentiable and convex problems. Unfortunately, the mixed-integer constrained optimization problems are just non-differentiable. However, the saddle point theorem [7] can provide sufficient conditions for solving constrained optimization problems without any differentiability or convexity requirements. It states that, if a point is a saddle point of the augmented Lagrange function associated with the primal problem, then the point is the solution of the primal problem. Accordingly, in this paper the saddle point theorem is used to solve mixed-integer constrained optimization problems.

The saddle point can be obtained by minimizing $L_a(\mathbf{z},\nu^*,\upsilon^*)$ with the optimal Lagrange multipliers (ν^*,υ^*) as a fixed parameter vector. However, the difficulty of this minimization is that it requires the knowledge of (ν^*,υ^*) previously. In general, the optimal values of Lagrange multipliers are unknown *a priori*. The duality theorem [7] can be employed to overcome this difficulty.

According to the duality theorem, we can construct an evolutionary max-min algorithm to solve mixed-integer constrained optimization problems. The evolutionary min-max algorithm (MIHDE-AMM) includes two phases as stated in Table 1. In the first phase (step 2 in Table 1), a mixed-integer evolutionary algorithm, called MIHDE [6], is used to minimize the augmented Lagrange function with multipliers fixed. In the second phase (step 3 in Table 1), the Lagrange multipliers are updated to ascend the value of the dual function toward obtaining maximization of the dual problem.

Table 1. Evolutionary max-min algorithm for mixed-integer constrained optimization problems.

Step 1. Set initial iteration: $l = 0$. Set initial multipliers: $\nu_k^l = 0$ for $k = 1,...,m_e$, $\upsilon_k^l = 0$ for $k = 1,...,m_i$. Set penalty parameters: $\alpha_k > 0$ for $k = 1,...,m_e$, $\beta_k > 0$ for $k = 1,...,m_i$.

Step 2. Use MIHDE to solve $L_a(\mathbf{z},\nu^l,\upsilon^l)$. Let $\mathbf{z}_b^l = (\mathbf{x}^l,\mathbf{y}^l)_b$ be a minimum solution to the function $L_a(\mathbf{z},\nu^l,\upsilon^l)$.

Step 3. Update the multipliers as follows:

$$\nu_k^{l+1} = h_k(\mathbf{z}_b^l) + \nu_k^l$$

$$\upsilon_k^{l+1} = \left\langle g_k(\mathbf{z}_b^l) + \upsilon_k^l \right\rangle_+$$

Step 4. Update α_k and β_k, if necessary.

Step 5. If the maximum iteration is reached, stop. Otherwise, let $l = l + 1$ and repeat Steps 2 to 4.

As far as the evolutionary computation is concerned, the evolutionary max-min procedure may increase many function evaluations and affect the convergence rate. However, in order to find the exact solution, it is necessary and inevitable unless using other special approaches, e.g., sequential quadratic programming (SQP) [8], to continuously update the Lagrange multipliers. Unfortunately, SQP is not applicable for the non-differentiable mixed-integer constrained optimization problems. The update of the Lagrange multipliers is based on the exact or approximate minimum of the augmented Lagrange function with multipliers fixed. As presented by Arora *et al.* [8], an exact or approximate minimum is necessary in order to ensure proper shift of the Lagrange function towards the required solution. With a rough minimum, the shift of the Lagrange function may be far away from the required solution leading to obtain a nonexistent dual function so that the duality theorem shall be disobeyed.

IV. Experimental Example

An example in Karimi [1] is given here to test the performance of the MIHDE-AMM algorithm. The MIHDE with penalty function method (MIHDE-PFM) is used to be compared.

In this example, we consider a three-unit, four-product production scheduling problem. Therefore, the MINLP problem with 8 equality constraints includes one real variable and 16 integer variables. Table 2 shows the data of processing times as discussed in Karimi [1]. Karimi obtained the minimum makespan of 34.8 hrs. In this example, we use the minimum value as a stopping criterion to check how much iteration for MIHDE-AMM reaching to this global solution. The product sequence of 1-3-4-2 with a makespan of 34.8 hrs, and the optimal completion times are shown in Figure 3.

Table 2. Processing times (h) of products.

Products	units		
	1	2	3
1	3.5	4.3	8.7
2	4.0	5.5	3.5
3	3.5	7.5	6.0
4	12.0	3.5	8.0

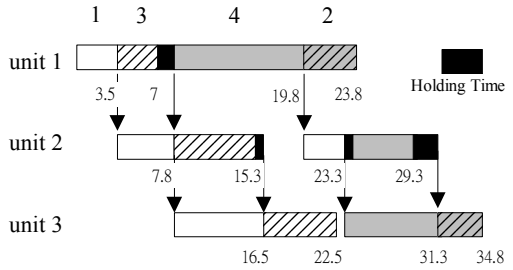

Figure 3. Optimal completion times for four products.

For comparison, The penalty function method (MIHDE-PFM) is also applied to solve the production scheduling problem. Table 3 shows that the computational results for MIHDE-AMM and MIHDE-PFM with penalty parameters of 1, 10^3 and 10^6. For penalty parameters of 10^3 and 10^6, both two algorithms can find the same optimal solution. However, the feasible solution are not obtained by MIHDE-PFM with penalty parameters of 1 because some inequality constraints are violated. Conversely, the MIHDE-AMM can obtain the same optimal solution using the penalty parameters of 1. This result demonstrates that the MIHDE-AMM outperforms the MIHDE-PFM.

IV. Conclusions

In this paper, a mixed-integer nonlinear programming (MINLP) model is developed to formulate the production scheduling problem. In order to effectively find the optimal solution, a mixed-integer evolutionary algorithm is proposed to solve this MINLP problem. From computational results, we can find that better results are obtained in comparison with the penalty function method. This demonstrates that the algorithm can effectively handle the decision-making problem of production scheduling.

Table 3 . Comparison of the MIHDE-AMM and MIHDE-PFM for various penalty parameters, α_k , $k = 1,\ldots,8$.

Item	MIHDE-AMM			MIHDE-PFM		
	$\alpha_k = 1$	$\alpha_k = 10^3$	$\alpha_k = 10^6$	$\alpha_k = 1$	$\alpha_k = 10^3$	$\alpha_k = 10^6$
$f(\mathbf{z})$	34.8	34.8	34.8	8.0†	34.8	34.8
$h_1(\mathbf{z})$	0.0	0.0	0.0	-1.0‡	0.0	0.0
$h_2(\mathbf{z})$	0.0	0.0	0.0	-1.0‡	0.0	0.0
$h_3(\mathbf{z})$	0.0	0.0	0.0	-1.0‡	0.0	0.0
$h_4(\mathbf{z})$	0.0	0.0	0.0	-1.0‡	0.0	0.0
$h_5(\mathbf{z})$	0.0	0.0	0.0	-1.0‡	0.0	0.0
$h_6(\mathbf{z})$	0.0	0.0	0.0	-1.0‡	0.0	0.0
$h_7(\mathbf{z})$	0.0	0.0	0.0	-1.0‡	0.0	0.0
$h_8(\mathbf{z})$	0.0	0.0	0.0	-1.0‡	0.0	0.0
TFC	21905	663	663	NTR	663	663

NTR: The global solution is not to reach.
†: The solution is infeasible.
‡: The constraint is violated.

References

[1] I. A. Karimi, "Multiproduct Batch Plant Scheduling in CACHE," Chemical Engineering Optimization Models with GAMS, vol. 6

[2] Z. Michalewicz, "Genetic Algorithm + Data Structure = Evolution Programs," Springer-Verlag, 1994.

[3] T. Back, D. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*. New York: Oxford Univ. Press, 1997.

[4] Z. Michalewicz, Z. and M. Schoenauer, "Evolutionary Algorithms for Constrained Parameter Optimization Problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1-32, 1996.

[5] Y. C. Lin, K. S. Hwang, and F. S. Wang, "An evolutionary Lagrange method for mixed-integer constrained optimization problems," *Engineering Optimization*, vol. 35, no. 3, pp. 267-284, 2003.

[6] Y. C. Lin, K. S. Hwang, and F. S. Wang, "A mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems," *Computers & Mathematics with applications*, vol. 47, pp. 1295-1307, 2004.

[7] D.A. Wismer and R. Chattergy, *Introduction to Nonlinear Optimization*. Elsevier North-Holland, 1978.

[8] J. S. Arora, A. I. Chahande, and J. K. Paeng, "Multiplier methods for engineering optimization," *Int. J. Numerical Methods in Engineering*, vol. 32, pp. 1485-1525, 1991.