Automatic Rule Combination Approach for Single-Stage Process Scheduling Problems

Yaohua He and Chi-Wai Hui
Chemical Engineering Dept., Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, P.R. China

DOI 10.1002/aic.11236
Published online July 2, 2007 in Wiley InterScience (www.interscience.wiley.com).

Meta-heuristic methods show better performance in solving large-size sequential process scheduling problems than mixed-integer linear programming. Heuristic rules are often used in meta-heuristic methods and play a very important role in reducing search space of the scheduling problems. In our previous work (He and Hui, Ind Eng Chem Res. 2006; 45: 4679–4692), approaches of how to automatically select rules from a set of heuristic rules have been proposed. This work proposes a novel approach of how to construct a comprehensive, but not very large set of rules according to the analysis of impact factors. Working with the new rule set, the original approaches are improved, in which a full rule sequence is used for schedule synthesis. A new automatic rule combination approach is proposed, in which a partial rule sequence is intentionally formed and used for schedule synthesis. The adoption of the partial rule sequences saves computational sources and increases the search ability of the algorithms. The automatic rule combination approach almost has the same search ability as the long time tabu search to find the near-optimal solutions to the large-size problems, but with much higher convergence speed. © 2007 American Institute of Chemical Engineers *AIChE J,* 53: 2026–2047, 2007

Keywords: sequential process scheduling, impact factors, heuristic rules, genetic algorithm, rule evolution

Introduction

Scheduling plays a crucial role in the efficiency of any production system. The process industry covers areas such as chemicals, oil refining, food, beverages, pharmaceuticals, and brewing. In contrast to the long-term concerns with scheduling in discrete parts manufacturing industry, the attention devoted to computer-aided scheduling methodology in the process industry is much more recent, beginning in the middle 1970s.[1,2] Process scheduling shows much more complexity than discrete machine scheduling.[3] Research on batch and continuous process scheduling has received great attention from academia and industry in the past two decades because of the increasing pressure to raise efficiency and reduce costs, the significant progress in related modeling and solution techniques, and rapidly developing computational power.[4]

Based on the complexity of processing sequences employed to produce products, all processes in multiproduct multipurpose plants can be classified into two different groups as follows.

Sequential processes

Different products follow the same processing sequence. It is usually possible to define processing stages, which can be single stage or multiple stages. There can be only one unit per stage or parallel units at each stage. For this type of

This article contains supplementary material available via the Internet at http://www.interscience.wiley.com/jpages/0001-1541/suppmat/.

Correspondence concerning this article should be addressed to C. W. Hui at kehui@ust.hk.

Table 1. Changeover Times, Due Dates, and Process Times of Example 1

	Changeover Times											Process Times		
	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	Due Date	u1	u2	u3
i1	–	1.02	0.89	0.80	0.98	1.24	1.83	0.84	1.58	1.02	31	14.07	12.20	5.40
i2	0.78	–	1.86	1.96	1.69	1.97	1.37	1.03	0.96	1.66	39	15.64	8.95	19.41
i3	1.66	1.87	–	1.69	1.88	0.81	1.43	0.78	1.49	0.63	22	16.49	14.29	8.56
i4	0.95	1.41	1.06	–	1.30	1.02	0.71	1.45	1.26	0.71	34	10.70	16.18	5.97
i5	0.72	1.44	1.68	1.81	–	1.26	1.17	1.74	0.77	0.74	55	15.53	5.13	10.68
i6	1.24	1.60	1.68	1.26	1.37	–	1.11	1.39	0.96	0.65	28	8.20	10.68	7.19
i7	1.96	1.99	1.85	0.81	0.70	1.13	–	1.83	1.34	0.75	55	14.41	17.24	5.73
i8	0.97	1.78	1.32	0.67	1.22	1.89	0.99	–	0.66	1.77	29	6.83	9.70	8.02
i9	1.07	1.81	0.71	1.57	0.78	1.83	1.34	1.43	–	0.54	26	5.08	6.43	10.41
i10	1.54	1.76	1.27	1.46	1.51	0.83	1.19	0.51	1.14	–	42	8.60	13.11	14.12

process, batches are used to represent production and it is thus not necessary to consider mass balances explicitly.

Network-represented processes

When production recipes become more complex and/or different products have low recipe similarities, processing networks are used to represent the production sequences. This corresponds to the more general case in which batches can merge and/or split and material balances are required to be taken into account explicitly. Kondili et al.[5] proposed a general framework of state-task network (STN) for the ambiguity-free representation of such processes. The main advantages and drawbacks of STN were summarized by Kallrath.[6] Pantelides[7] then proposed an alterative representation, the resource-task network (RTN), which describes processing equipment, storage, material transfer, and utilities as resources in a unified way.

Single-stage multiproduct scheduling problems (SMSP) in batch plants with parallel units are a typical kind of sequential scheduling problems in process industry. An extension of SMSP is the multistage multiproduct scheduling problems (MMSP) in batch plants with parallel units. The main solution methods for these scheduling problems can be mainly summarized as mathematical programming (MP) (including mixed-integer linear programming (MILP), mixed-integer nonlinear programming (MILNP), and constraint programming (CP)), list scheduling (using scheduling rules), random search plus heuristic rules, and meta-heuristic methods.

MP has been widely studied by researchers in academia. The procedure of MP is as follows: establish the MILP or MINLP model first, including objective function and constraints, and then solve the model using a modeling software (like GAMS) with embedded commercial solvers such as OSL, CPLEX, XPRESS, and DICOPT.

Pinto and Grossmann[8] presented a continuous time MILP model for MMSP. They used the concept of parallel time coordinates for units and tasks. Furthermore, Pinto and Grossmann[9] proposed an alternative model in which the preordering of orders was imposed explicitly, by applying an alternative representation of the time slots for the units. This resulted in a significant reduction in computational time. Cerda et al.[10] proposed a continuous time MILP model for SMSP. They used tri-index decision variables as well as the concept of predecessor and successor to describe the order assignment to various production units while taking into account sequence dependent changeover constraints. To deal with large-size problems, they proposed heuristics, such as the order preordering, to reduce the number of feasible predecessors for each order. On the basis of the notation of time slot, Karimi and Mcdonald[11] proposed two models for parallel semicontinuous processes considering the sequence-dependent setup times, orders, and their corresponding due dates in order to minimize the inventory. The major advantage of the formulation is that it can incorporate fixed time events such as due dates while using continuous representation of time.

Hui and Gupta[12] presented a general formulation for SMSP. The proposed formulation applies three sets of bi-index variables to handle order sequence-dependent constraints either with or without imposing preordering heuristics. The main advantage of this formulation compared to the other previously proposed formulations is the significant reduction in the number of binary variables and consequently shortening the solution time. The authors claimed their method suitable for handling large-size industrial problems.

Table 2. Results of Example 1 by MILP

Original Model				Improved Model (n = 4, Big M = 150)			
Iterations	Nodes	CPU Time, s	Makespan	Iterations	Nodes	CPU Time, s	Makespan
592		No integer solution yet		844		No integer solution yet	
1000	46	0.27	36.63	1000	27	0.22	47.29
2000	81	0.50	36.63	2000	56	0.42	37.53
20,000	769	4.53	31.74	20,000	596	4.01	37.53
200,000	6896	45.64	31.53	200,000	5042	39.46	28.31
2,000,000	77,642	768.05	28.79	2,000,000	56,843	542.78	28.31
3,000,000	118,249	1176.95	28.79	3,000,000	88,667	1044.68	28.31

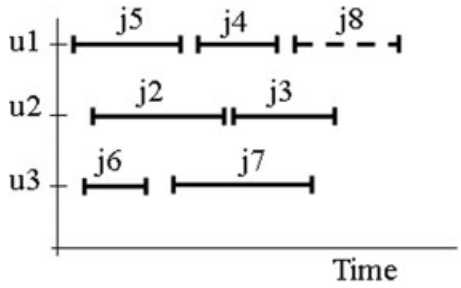

Figure 1. Illustration of PsT.

Chen et al.[13] developed a MILP model for SMSP using the continuous-time representation and the notation of time slot. When the model was developed, the allocation of orders and units to time slots was represented by two sets of binary variables. The MILP model not only involved fewer binary variables than any other models based on the notation of time slot,[14] but also could be used to optimize several types of objective functions.

An important recent advance was the introduction by Sundaramoorthy and Karimi[15] of a formulation without big-*M* constraints that proved more efficient than other competing methods for both maximizing profit and minimizing makespan.

Castro and Grossmann[16] presented a new RTN-based continuous-time MILP model for SMSP. It was based on the general formulation of Castro et al.,[17] but had one important difference: a different time grid was used for each unit instead of a single time grid for all events taking place. New constraints were presented that allowed the consideration of both release and due dates in a general way. The new formulation was shown to perform much better than other continuous-time MILP formulations and standalone CP models,[18] and was comparable to the hybrid MILP/CP algorithm of Maravelias and Grossmann[19] on a set of example problems where the objective was the minimization of total cost, or the minimization of total earliness. However, this model did not consider changeovers between products. Hence, Castro et al.[20] later presented two new multiple-time-grid continuous-time formulations for SMSP and MMSP, where equipment units are subject to sequence-dependent changeovers, and product orders are subject to both release and due dates, where the objective was the minimization of total cost, total earliness, or makespan. For makespan minimization, the multiple-time-grid, continuous-time formulation by Castro et al.[16,20] is not the best efficient MP method. The CP method and the discrete-time formulation were found to be the best MP methods.[21] However, these two methods have the disadvantage of considering integer data and show a rapid decrease in performance with an increase in problem complexity.

CP is another solution approach that can be used for solving some classes of scheduling problems. CP and MILP have complementary strengths that can be combined into hybrid algorithms, yielding considerable computational improvements when compared with the standalone approaches. Examples of these are the work of Jain and Grossmann[18] for single-stage, Harjunkoski and Grossmann[22] for multistage multiproduct plants.

Further details on the aforementioned and other available approaches for short-term scheduling can be found in the recent review papers of Floudas and Lin[4] and Mendez et al.[23] MP is able to obtain optimal solutions to small-size scheduling problems. However, as the problem size increases linearly, the computational time of MP will increase exponentially. It is very difficult for MP to obtain an acceptable solution to large-size problems within reasonable time. In most studies, authors used small-size problems to illustrate their methods. Therefore, the mathematical methods have mainly theoretical, but not practical or applicable significance.

To solve large-size scheduling problems, due to the difficulties of the MP, the preferred method in industry is to use scheduling rules, such as the shortest processing time first (SPT) rule and the earliest due date first (EDD) rule. According to a scheduling rule, jobs are sequenced in decreasing priority order and then one by one assigned to machines or processing units. During the last 30 years, the performance of a large number of scheduling rules has been studied extensively using simulation techniques. The research on the scheduling rules has shown that there is no single universal rule, and the effectiveness of a scheduling rule depends on the scheduling objective and the prevailing shop or plant conditions. Cheng and Sin[24] presented a full review of parallel machine scheduling research, including a large number of scheduling rules. Park et al.[25] investigated the significant scheduling rules for parallel machine scheduling.

In the research on process scheduling, heuristic rules are often combined into the MILP models with the purpose of reducing the size of the models.[9,10,12,13]

However, due to constraints inherent in some scheduling problems, the simple rule-based method may not guarantee the feasibility and optimality of the solution. Mokashi and Kokossis[26] pointed out that a general-purpose optimization approach resorts to using conventional MP techniques on generic models of a scheduling problem, which is limited when applied to large-scale industrial problems because of the computational time involved. The other extreme of heuristic methods lacks guarantees of the quality of the solution. Therefore they proposed a philosophy of contextual optimization that exploited problem-specific knowledge to develop efficient algorithms. This concept was applied to a delivery scheduling problem to generate a tailored graph-based method called the maximum order tree algorithm, which, compared to conventional methods, reduced the CPU time dramatically without compromising solution quality. When applied to a single-site distribution case study, it resulted in savings of over a quarter of a million dollars per year over the existing heuristic-rule based system.

Table 3. Calculation of Simple/Compound Rules

Number of Impact Factors	$n_s + n_c$
2	$C_2^1 + C_2^2 = 3$
3	$C_3^1 + C_3^2 + C_3^3 = 7$
4	$C_4^1 + C_4^2 + C_4^3 + C_4^4 = 15$
5	$C_5^1 + C_5^2 + \cdots + C_5^5 = 31$
...	...
n_f	$C_{n_f}^1 + C_{n_f}^2 + \cdots + C_{n_f}^{n_f}$

Table 4. Seven Rules for Minimization of the Makespan-Related Objectives in SMSP

Rule	Detailed Content	Shortened Form	Factors Involved	Simple or Compound
Rule 1	Assign the order on the unit that makes the order's possible start time be as early as possible, that is, assign the order on the first available unit (FAU)	Earliest possible start time, FAU	Possible start time (PsT)	Simple
Rule 2	Assign the order on the unit that makes the order's changeover time on the unit be the shortest	Shortest changeover time, SCT	Changeover time (CT)	Simple
Rule 3	Assign the order on the unit that makes the order's process time on the unit be the shortest	Shortest process time, SPT	Process time (PT)	Simple
Rule 4	Assign the order on the unit that makes the order's start time be as early as possible	Earliest start time, EST	Possible start time and changeover time	Compound
Rule 5	Assign the order on the unit that makes the sum of the order's possible start time and process time on the unit be the shortest	Shortest possible start time + process time, SPsPT	Possible start time and process time	Compound
Rule 6	Assign the order on the unit that makes the sum of the order's changeover time and process time on the unit be the shortest	Shortest changeover time + process time, SCPT	Changeover time and process time	Compound
Rule 7	Assign the order on the unit that makes the order's completion time be as early as possible	Earliest completion time, ECT	Possible start time, changeover time, and process time	Compound

In random search, the tasks or products are randomly sequenced, and then one by one assigned to machines or processing units. Through exploring a set of random solutions, feasible solutions better than the simple rule-based method can be attained. Our research group[27] has proposed a random search based on heuristic rules, which outperforms MILP in solving large-size scheduling problems for a single-stage batch plant with parallel units.

In recent years, meta-heuristics, such as genetic algorithm (GA), simulated annealing (SA), and tabu search (TS), and their hybrids have shown potential to solve large-size complex machine scheduling problems. Tanev et al.[28] developed a hybrid evolutionary algorithm capable of scheduling 400 customers' orders. Nowicki and Smutnocki[29] proposed an algorithm based on a TS technique to solve a permutation flow shop problem (up to 500 jobs and 20 machines). Grabowsk and Wodecki[30] proposed a new very fast local search procedure based on a TS approach for the permutation flow shop problem with makespan criterion. They solved a 500-job/20-machine flow-shop problem with high accuracy in a very short time.

Due to the complexity and constraints of process scheduling, some authors believe that meta-heuristics are not suitable for process scheduling problems, especially large-size complex problems. Nevertheless, our research[31] has shown that meta-heuristic methods, combined with suitable heuristic rules, are effective to search near-optimal solutions for large-size process scheduling problems. For example, GA can be used to evolve a set of random feasible solutions and obtain much better solutions than the random search. Moreover, we have put forward two rule evolutionary approaches (meta-heuristic plus heuristic rules): one is an automatic rule selection (ARS) method, and another is a rule sequence evolution (RSE) method.[31]

In our work, ARS method was based on GA, in which mixed chromosomes were adopted. A mixed chromosome consists of a rule sequence and an order sequence. In each mixed chromosome, the rule at the head of the rule sequence is used to assign all of the orders in the order sequence to the units. At the end of computation, a best rule is selected automatically.

RSE method was also based on GA, in which mixed chromosomes were still utilized. The algorithm tries to use a rule to assign an order to the one unit, starting from the head of the rule sequence. If a rule failed to assign an order, the next rule would be tried, until the order is assigned successfully. If no rule to be used successfully, the chromosome is infeasible, then a new chromosome will be regenerated. At the end of computation, a best rule sequence is evolved automatically. ARS and RSE are self-learning approaches. By using these two approaches, the algorithms themselves will finally choose the suitable rule or rule sequence to synthesize the evolved order sequence into a high quality schedule.

To sum up, in our study to solve the process scheduling problems, e.g. scheduling problem of single-stage batch plant with parallel units, by using random search or GA combined with profixed rule, or by using ARS or RSE methods, the heuristic rules played very important roles in cutting down search space and increasing search speed. In all these methods, heuristic rules were selected from a candidate rule base, which was constructed randomly just for the illustration of the methods. Now that the rules are so important to algorithms, we should construct the rule base scrupulously, not

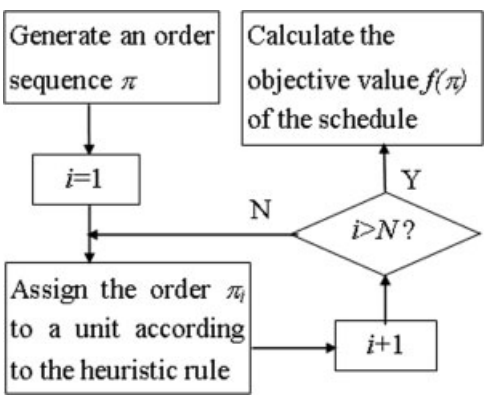

Figure 2. Schedule synthesis by one rule.

Table 5. A Random Order Sequence Scheduled by Different Rules

Order Sequence π	3 2 7 6 4 5 9 10 1 8						
Rule Used	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7
Makespan C_{max}	38.24	58.86	37.62	38.24	31.94	30.72	30.14
Total tardiness T	10.46	47.86	6.62	10.46	2.94	1.72	0.09
Total flow time F	224.13	260.46	182.79	224.13	184.82	176.89	197.72
Compound objective TC	48.70	106.72	44.24	48.70	34.88	32.44	30.23
Compound objective TF	234.59	308.32	189.41	234.59	187.76	178.61	197.81

letting some possible useful rules lie out of our consideration. In practice, the rule base can be enlarged if necessary. In the ARS and RSE methods, each rule needs a subroutine. When the rule base is large, the workload to write code will be huge. Furthermore, managing a large rule base is also a tough task. Hence we propose a rule combination method based on impact factors analysis. This study proposes a rule combination method for specific scheduling objectives, e.g. makespan-related objectives. Automatic rule combination (ARC) methods are also proposed and studied.

SMSP is used to illustrate the rule combination approach. SMSP has been widely used by process scheduling researchers, such as Cerda et al.,[10] Hui and Gupta,[12] and Chen et al.[13] The proposed approaches can be applied to other sequential process scheduling problems, e.g. MMSP.

The algorithms in this paper were implemented in C language and the computation tests were run on a computer with an Intel Pentium M 1500 MHz CPU and 768 MB of memory. GA is just for illustration, not the only choice. Other meta-heuristics, such as TS, and hybrid algorithms, may be more effective in solving large-size problems.

Problem Definition

In SMSP, a fixed number of production units (forming a set of units: U) are available to process all customer orders (forming a set of orders: O). Assume the number of production units is M, and the number of customer orders is N. Each order involves a single product, requiring a single processing stage, has a predetermined due date, and can only be processed in a subset of the units available. The production units have different processing capacities. Hence, the process time of the same order is fixed and production unit dependent. A production unit processes only one order at a time. When one order changes over to another order, time is required for the preparation of the unit for the changeover. The changeover time is sequence-dependent. Forbidden changeovers and processes, called CP constraints, may exist in the problem.

Time-based scheduling objectives are commonly considered in the literature because cost-based objectives usually can be surrogated by time-based objectives. Common time-based objectives are described by Pinedo.[32] Makespan, total tardiness, total earliness, and total flow time are four typical scheduling objectives to be minimized. Assume that there are N orders to be assigned to M units.

1. *Makespan (C_{max}):* The makespan, defined as:

$$C_{\max} = \max\{C_1, C_2, \ldots, C_N\} \tag{1}$$

is the completion time of the last order to leave the system, where C_j is the completion time of order j. A minimum makespan usually implies a high utilization of the units.

2. *Total tardiness (T):* The total tardiness is the sum of the tardiness of all orders, defined as:

$$T = \sum_{j=1}^{N} T_j \tag{2}$$

where T_j is the tardiness of order j: $T_j = \max\{C_j - d_j, 0\}$, d_j is the due date of order j.

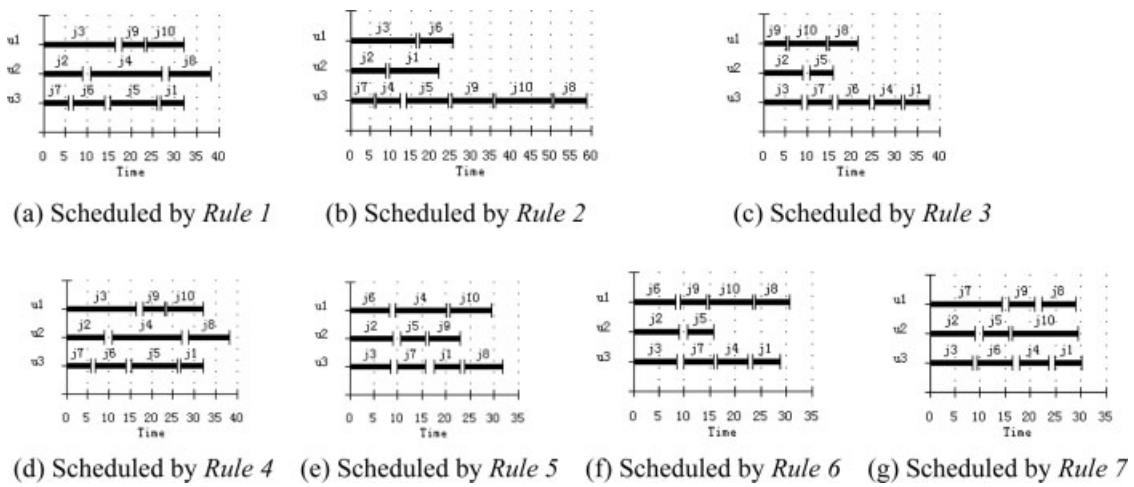

Figure 3. Different schedules for the same order sequence by different rules.

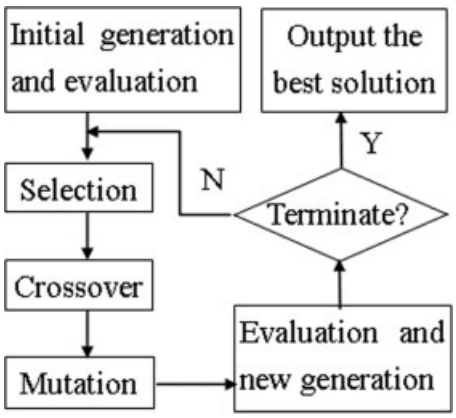

Figure 4. Flow chart of GA.

3. *Total earliness (E):* The total earliness is the sum of the earliness of all orders, defined as:

$$E = \sum_{j=1}^{N} E_j \tag{3}$$

where E_j is the earliness of order j, $E_j = \max\{d_j - C_j, 0\}$.

4. *Total flow time (F):* The total flow time is the sum of completion time of all orders:

$$F = \sum_{j=1}^{N} C_j \tag{4}$$

5. The minimization of the makespan and the minimization of the total tardiness can be conflicting objectives. If the due dates of the orders are loose, it is easy to find a schedule with zero total tardiness, but the makespan of the schedule may be still very long. To consider both makespan and total tardiness at the same time, we propose a new *compound objective TC*:

$$TC = \alpha T + \beta C_{\max} \tag{5}$$

where T is the total tardiness, and $C_{\max}$ is the makespan.

6. Similarly, to consider both total flow time and total tardiness simultaneously, we put forward a new *compound objective TF*:

$$TF = \sum_{j=1}^{N} (\alpha T_j + \beta C_j) \tag{6}$$

where T_j is the tardiness of order j: $T_j = \max\{C_j - d_j, 0\}$, d_j is the due date of order, and C_j is the completion time of order j . α and β are weight coefficients. We let $\alpha = \beta = 1$.

Here we classify the scheduling objectives into two categories: makespan-related objectives (such as makespan, total tardiness, total flow time, and compound objectives *TC* and *TF*) and earliness-related objectives. For the former, we try to let the orders be completed as early as possible. For the later, we try to let the orders complete as near as possible to the due date, but not after the due date. In this paper, we focus on the makespan-related objectives. Just as done in the literature,[10,12] when minimizing the makespan of a problem, the due dates of the orders are not considered as constrains. If due dates are considered as constraints when minimizing makespan, the complexity of MILP model increases. However, in our work of GA, it is just an option, not an actual limitation, that the due dates are not considered as constraints when minimizing makespan. Actually, adding a penalty term in the objective function for the orders violating due dates enable the algorithm to find a good schedule without tardy orders, assuming that there are no tardy orders at optimum. In case of tardy orders at optimum, soft due dates[33] should be used, which is determined by minimizing the tardiness, and hence increasing computational effort. Therefore, we do not consider due dates as constrains when minimizing makespan-related objectives in this work.

Example 1 will be used to illustrate our approach. Example 1 is a random SMSP in which 10 orders ($N = 10$) are to be assigned to three units ($M = 3$). Each order involves only one batch, required to be completed before the due date d_i. The due date d_i is produced randomly, $d_i \in (20, 60)$. All the order release times and unit release times are null. Changeover time c_{ij} is produced randomly, $c_{ij} \in (0.50, 2.00)$; and process time p_{iu} is also produced randomly, $p_{iu} \in (5.00, 20.00)$. Neither forbidden changeover nor forbidden process exists in this example. The data for Example 1 is presented in Table 1.

MILP Model and Its Difficulties

For SMSP, Cerda et al.[10] did the original work with MILP model, later Karimi and McDonald,[11] Hui and Gupta,[12] and Chen et al.[13] presented their modified MILP models. All these works focused small problems, and the later improvements were marginal, not substantial, especially for large-size problems. So the MILP model by Hui and Gupta,[12] a bi-index MILP model based on continuous-time representation, is representative, maybe not very efficient. The bi-index model is applied to Example 1, being formulated in GAMS

Table 6. Results of Example 1 by GA Combined with Different Rules (min $C_{\max}$)

Method	Best $C_{\max}$	Mean $C_{\max}$	Mean Dev. from Optimum	Mean CPU Time, s	Mean Iterations	Tests	Rule Quality
GA_R1 (FAU)*	28.31	28.36	0.17	0.0715	18.9	10	Second
GA_R2 (SCT)	28.31	29.85	5.44	0.066	19.2	10	Third
GA_R3 (SPT)	36.31	36.31	28.26	<0.055	6.4	10	Worst
GA_R4 (EST)	28.31	28.36	0.17	0.077	18.5	10	Second
GA_R5 (SPsPT)	28.31	28.31	0.00	0.055	8.9	10	Best
GA_R6 (SCPT)	30.53	30.53	7.84	<0.055	6.0	10	Fourth
GA_R7 (ECT)	28.31	28.31	0.00	0.055	9.2	10	Best

*GA_R*n* stands for GA combined with Rule *n*.

Table 7. Results of Example 1 by GA Combined with Different Rules (min F)

Method	Best F	Mean F	Mean Dev. from Optimum	Mean CPU Time, s	Mean Iterations	Tests	Rule Quality
GA_R1 (FAU)*	75.35	75.98	3.53	0.072	22.5	10	Worst
GA_R2 (SCT)	73.55	73.76	0.50	0.055	21.4	10	Second
GA_R3 (SPT)	73.39	73.39	0.00	<0.055	10.8	10	Best
GA_R4 (EST)	75.35	75.89	3.40	0.061	23.5	10	Fourth
GA_R5 (SPsPT)	75.35	75.35	2.67	0.055	17.4	10	Third
GA_R6 (SCPT)	73.39	73.39	0.00	<0.055	10.5	10	Best
GA_R7 (ECT)	75.35	75.35	2.67	0.055	17.4	10	Third

*GA_Rn stands for GA combined with Rule n.

and solved by OSL, and the results are presented in Table 2 under the original model.

Actually, the single-stage problem can be regarded as an irregular assignment problem in which the orders to be processed by each unit can be limited under a certain number n. The value of n can be chosen through logical estimation and a "trial and error" method. This is in all aspects similar to the well-known method of searching the global optimal solution over different values of the number of event points in time grid based, continuous-time formulations. Furthermore, the value of Big M also influences the performance of the model. Theoretically, the value of the Big M is not usually so critical in the MILP models performance especially in larger-sized problems. A rough approximation is often made around the approximated scheduling horizon. Models relying on the use of Big M comprise other disadvantages such as a larger integrality gap while solving. However, computational experiments show that a proper value of Big M has heavier effect on the performance of the model studied. If the Big M is selected a suitable value, much computational effort can be saved in obtaining a better solution. In the bi-index MILP model[12] and the tri-index model,[10] these two factors were not under consideration. Therefore, besides the constraints in the bi-index model, another constraint is given as follows:

$$\sum_{i=1}^{N} W_{iu} \leq n \quad (n \leq N) \tag{7}$$

where W_{iu} is a binary variable to express the assignment of order i to unit u. If order i is assigned to unit u, then $W_{iu} = 1$; else, $W_{iu} = 0$. The results of Example 1 by the improved model are also presented in Table 2.

As for the MILP model, our computational experiments show the following phenomena or difficulties:

1. The MILP model has to run for a certain number of iterations before it finds a feasible solution. For instance, in solving Example 1, at iteration 592 by the original model, and at iteration 844 by the improved model, the GAMS showed "No integer solution yet." In solving more large-size problems by MILP, it takes much more iterations (or CPU time) to get a feasible solution.

2. At the early stage of the computation, solution quality increases quickly. However, after running for some time, it often takes relatively long time to get another better solution. Example 1 is a small-size problem, but even at iteration 3,000,000 which takes about 20 min, the best solution with a makespan 28.31 is not achieved by the original model. With the problem size increasing, the search time to find a "good enough" solution is getting unbearable.

3. Theoretically, MILP can find the optimal solution to a problem if given enough time. However, even if the best solution is found, a lot effort or time may be required to prove optimality. For some small-size problems, the optimum may be found within a small number of iterations. In solving Example 1 by the improved model, the makespan 28.31 was obtained at iteration 26,149, but until iteration 3,000,000, the optimum had not been affirmed. If the problem size increases, the search time to affirm the optimum is getting more unbearable.

4. After the basic model has been established, adding some constraints (e.g. Eq. 7) or changing the parameters in the existing constraints (e.g. the Big M) makes it possible to get much better solutions within shorter time. To do this, a "trial and error" method has to be used. A lot of time is needed to find the suitable parameters in the constraints. Hence simulation experiments are required to refine the MILP model.

In the literature,[10,12] a heuristic is adopted to reduce search space—orders assigned to a unit are processed in a sequence of increasing due dates. By this way, search time is reduced, but the optimum of the original problem cannot be guaranteed.

In the following sections of this paper, heuristic and meta-heuristic methods are studied. These methods can easily obtain "good enough" solutions within reasonable computational time, although they cannot prove the optimality. To evaluate the performance of the algorithms, the following criteria are used: (1) the best solution that the algorithm obtains; (2) computational time; (3) the mean objective value of the computational tests; (4) the deviation from the best solution obtained up to now. For large-size problems, we

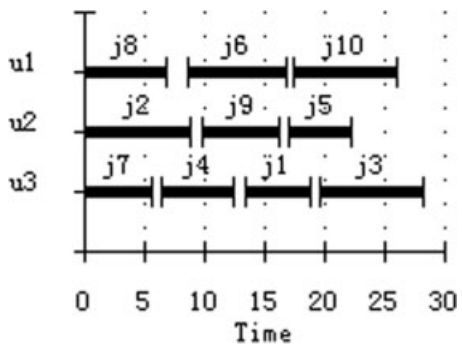

Figure 5. An optimal schedule of Example 1.

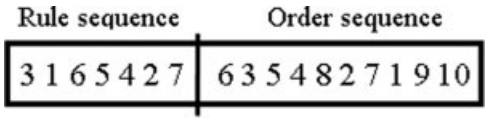

Figure 6. A mixed chromosome in ARS.

cannot expect to find globally optimal solutions within a reasonable time. Therefore, relative deviations from the best solutions are used as the criteria for evaluation.[34,35] Relative deviation is calculated with respect to the best (or optimal) solution obtained up to now. For makespan, we have:

$$\text{Dev. from best}(\%) = 100 \times [(\text{algorithm makespan} - \text{best makespan})/\text{best makespan}] \quad (8)$$

Rule Construction Based on Impact Factors Analysis

Due to difficulties with MILP in solving large-size scheduling problems, other alternative approaches can be applied, such as CP and MILP/CP,[19] list scheduling (using scheduling rules), random search plus heuristic rules,[27] and meta-heuristic methods.[31] In our previous work on random search and GA combined with heuristic rules, the heuristic rules were selected from a candidate rule base, which was randomly constructed just for illustration of the methods. Since heuristic rules are important to reduce search space, we should construct the rule base scrupulously and not excluding possibly useful rules. In practice, the rule base can be enlarged if necessary, according to our experience and knowledge. In the algorithms, each rule needs a subroutine. When the rule base is large, the workload to write code will be huge. Furthermore, to manage a large rule base is also a tough task. So it is desirable that a comprehensive, but not large, rule base is constructed. This is the reason why we propose a rule combination method based on impact factors analysis. We hope to establish an efficient, considerate but not too large rule base for a category of scheduling objectives. This paper is limited to the rule combination method for makespan-related objectives.

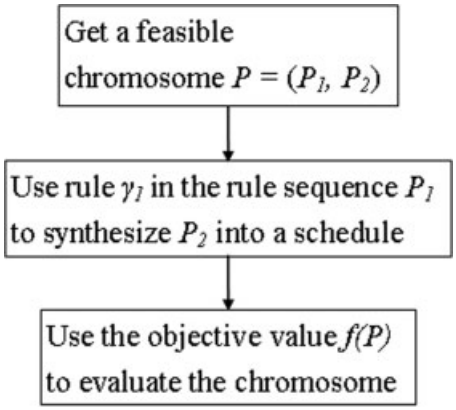

Figure 7. Evaluation procedure in ARS1.

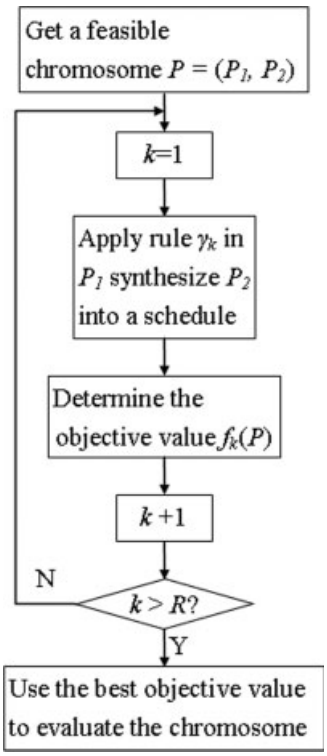

Figure 8. New evaluation procedure in ARS2.

Impact factors analysis

Impact factors are the basic parameters or variables of each order, which impact on objective value. If a scheduling objective is time-based, then the factors impacting on objective value are also time-based. For example, in SMSP, for minimization of the makespan-related objectives, the factors that impact on the objective value are the possible start time (PsT), changeover time (CT), and process time (PT) of each order. CT and PT are easily understood from Table 1. What is the PsT?

Let us use Figure 1 to explain the PsT. Assume that we have an order sequence π = (6, 2, 5, 4, 3, 7, *8*, 1, 9, 10), and the former six orders (6, 2, 5, 4, 3, 7) have been assigned to the units, order 8 is to be assigned now. Order 8 has three PsT, the completion times of order 4, 3, and 7. Order 8 may be assigned following order 4, 3, or 7. Even if order 8 is decided to follow order 4, the completion time of order 4 is still PsT of order 8, but not the start time (ST), because there is a CT between order 4 and 8. The ST is equal to the PsT plus the CT. It can be easily found that the PsT is order sequence dependent and unit dependent.

If unit setup time (UsT) is required when starting a processing of an order (as in MMSP),[33,36] then the UsT of an order becomes another impact factor on scheduling objective. All of these factors are time-based. Under certain circumstance, two or more factors can be united to one factor. For instance, if the UsTs and PTs of the orders are all unit dependent, then the factor UsT and PT can be composed together as a united factor. The adoption of a united factor subsequently simplifies the factor combination and rule combination. Assume that the number of the impact factors

Table 8. Results of Example 1 by ARS (min C_{max})

	ARS1				ARS2			
Test	C_{max}	CPU Time, s	Iterations	Rule Selected	C_{max}	CPU Time, s	Iterations	Rule Selected
1	28.31	0.055	8	Rule 5	28.31	0.110	7	Rule 7
2	28.31	0.055	10	Rule 5	28.31	0.160	9	Rule 7
3	28.31	0.055	10	Rule 7	28.31	0.160	8	Rule 7
4	28.31	0.055	8	Rule 5	28.31	0.160	9	Rule 7
5	28.31	0.055	9	Rule 5	28.31	0.160	8	Rule 5
6	28.31	0.055	8	Rule 5	28.31	0.160	9	Rule 7
7	28.31	0.055	11	Rule 7	28.31	0.110	8	Rule 7
8	28.31	0.055	9	Rule 5	28.31	0.220	11	Rule 7
9	28.31	0.055	13	Rule 5	28.31	0.160	8	Rule 7
10	28.31	0.055	11	Rule 7	28.31	0.160	9	Rule 7
Mean	*28.31*	*0.055*	*9.7*		*28.31*	*0.160*	*8.6*	

considered in the problem is n_f. In this paper, we use the instances with $n_f = 3$ to illustrate our approach.

Rule combination

The purpose of rule combination is to find all the useful but not redundant rules for a category of scheduling objectives. When using the rules we have two assumptions: (1) Any time when we consider a rule, we mean that the rule must involve one certain factor or more; (2) We only consider favorable rules. Under these two assumptions, a simple rule is defined as a rule that involves only one of the impact factors of the problem. What is the relationship between a simple rule and a factor? In general, a factor corresponds to two simple rules: one is in favor of the selected scheduling objective; another is an unfavorable rule. For example, SPT (shortest process time first) is a simple rule in favor of minimization of the makespan-related objectives; but LPT (longest process time first) is the unfavorable one. When we mention the simple rules later, we refer to the favorable ones. Based on this assumption, each factor corresponds to one simple rule as shown in Table 4. Hence, the number of the simple rules, n_s equals to the number of the factors n_f:

$$n_s = C_{n_f}^1 = n_f, \tag{9}$$

where $C_{n_f}^1$ represents the number of the combinations of n_f elements taken one at a time.

A compound rule is defined as a rule that involves two or more factors. In a compound rule, two or more factors are considered simultaneously. For instance, as stated previously (see Figure 1), the ST is equal to the PsT plus the CT, hence the rule EST (earliest start time) involves the two factors, PsT and CT. When we mention the compound rules later, we also refer to the favorable ones. If there are n_f factors considered, then the number of compound rules is n_c:

$$n_c = C_{n_f}^2 + C_{n_f}^3 + \cdots + C_{n_f}^{n_f} \tag{10}$$

Table 3 shows a rule calculation based on impact factors. In SMSP, for minimization of makespan-related objectives, the main three impact factors are PsT, CT, and PT. Considering these three impact factors, seven unit selection rules are summarized in Table 4. Rules 1–3 are simple rules, and Rules 4–7 are compound rules.

Performance of different rules

Natural numbers 1, 2, 3, . . . , N are used to denote N orders in SMSP. An order sequence $\pi = (\pi_1, \pi_2, \pi_3, \ldots, \pi_N)$ is produced randomly, $\pi_i \in \{1, 2, 3, \ldots, N\}$, $i = 1, 2, 3, \ldots, N$. And then, from π_1 to π_N, one by one, each order will be assigned to the units according to a certain heuristic rule above. As a result, a schedule is formed with an objective value, $f(\pi)$. $f(\pi)$ can be calculated by the functions 1–6.

Figure 2 is the procedure to synthesize an order sequence into a schedule according to one selected rule.

Assume that a random order sequence is π = (3, 2, 7, 6, 4, 5, 9, 10, 1, 8) in Example 1. The above seven rules are used, respectively, to assign all the orders in the order sequence to the units, and different schedules with various

Table 9. Results of Example 1 by ARS (min F)

	ARS1				ARS2			
Test	F	CPU Time, s	Iterations	Rule Selected	F	CPU Time, s	Iterations	Rule Selected
1	73.39	0.110	11	Rule 6	73.39	0.220	13	Rule 6
2	73.39	0.110	12	Rule 3	73.39	0.160	11	Rule 6
3	73.39	0.110	13	Rule 3	73.39	0.160	8	Rule 6
4	73.39	0.110	13	Rule 6	73.39	0.160	10	Rule 6
5	73.39	0.110	12	Rule 3	73.39	0.160	12	Rule 6
6	73.39	0.110	17	Rule 6	73.39	0.220	13	Rule 6
7	73.39	0.110	15	Rule 3	73.39	0.160	11	Rule 6
8	73.39	0.110	14	Rule 6	73.39	0.160	8	Rule 6
9	73.39	0.110	13	Rule 3	73.39	0.160	11	Rule 6
10	73.39	0.110	14	Rule 6	73.39	0.160	11	Rule 6
Mean	*73.39*	*0.110*	*13.4*		*73.39*	*0.172*	*10.8*	

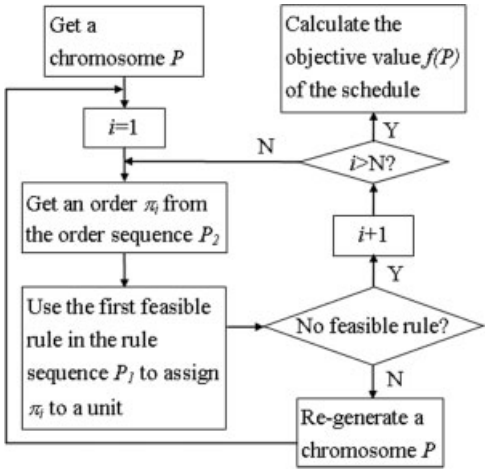

Figure 9. Evaluation procedure in RSE1.

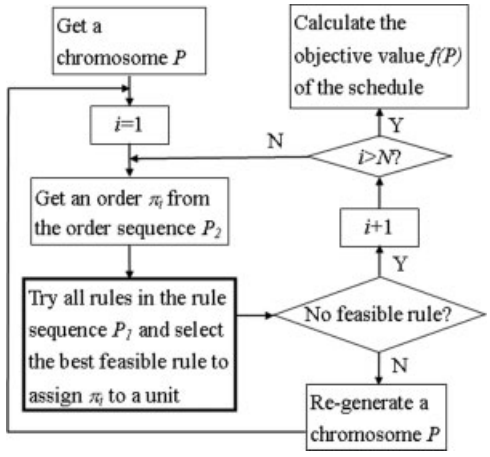

Figure 10. New evaluation procedure in RSE2.

objective values are formed (see Table 5 and Figure 3). For C_{max}, T, and TC, Rule 7 is the first best rule, Rule 6 the second. But for F and TF, Rule 6 is the first best rule, Rule 5 the second. For the same order sequence, when different rules are used for assignment, different quality schedules are obtained.

Simulation Experiments of GA Combined with Different Rules

GA procedure

When GA is applied to solve a sequential scheduling problem, solutions to the problem are represented by chromosomes. Permutation-based representation is adopted in our work, which is a kind of integer coding. As stated previously, natural numbers 1, 2, 3, . . . , N are used to denote N orders. A random order sequence $\pi = (\pi_1, \pi_2, \pi_3, \ldots, \pi_N)$ is produced, $\pi_i \in \{1, 2, 3, \ldots, N\}$. π is called a chromosome in GA. For example, π = (6, 3, 5, 4, 8, 2, 7, 1, 9, 10) is a sample chromosome of the 10-order problem in Example 1. The evaluation of a chromosome is a process to synthesize the chromosome (according to a preselected heuristic rule) into a schedule with an objective value. The process is shown in Figure 2. Traditionally, only one rule is applied for the evaluation of all the chromosomes in the GA process.

At the beginning of GA, an initial generation of chromosomes is produced randomly. Assume the number of chromosomes in the initial generation is *popsize*, which depends on the problem size and is an important parameter in GA for controlling the solution quality. In all the GAs in this paper, popsize = 200. At every iteration of GA, a new generation will be produced through crossover, mutation, and selection, and popsize will remain constant. Throughout genetic evolution, because of the mechanism of selection, crossover, and mutation, good-quality offspring are born from the previous generation (parents). Generation by generation, stronger chromosomes are the survivors in a competitive environment. At the end of GA, optimal or near-optimal solutions can be achieved. The detailed components of GA were described in the paper.[31] The procedure of GA is shown in Figure 4.

Assume the number of the chromosomes that are selected from a generation to crossover is *xsize*; the number of the chromosomes that are selected from the generation to mutate is *msize*. We can let xsize + msize = popsize. The ratio C_r = xsize/popsize is called crossover rate, often $C_r \in [0.5, 0.9]$; and the ratio M_r = xsize/popsize called mutation rate, often $M_r \in [0.1, 0.3]$. So $C_r + M_r = 1$. C_r and M_r are two important parameters that influence the convergent performance of GA. In general, if M_r increases, GA converges slowly on a final solution, and thus, GA has more chances to arrive at better solutions. But if M_r is too large, GA tends to be like

Table 10. Results of Example 1 by RSE (min C_{max})

	RSE1				RSE2			
Test	C_{max}	CPU Time, s	Iterations	Rule Sequence P_1	C_{max}	CPU Time, s	Iterations	Rule Used to Assign π_i
1	28.31	0.110	10	7 6 4 3 1 5 2	28.31	0.160	9	Rule 7 or 4
2	28.31	0.110	9	7 3 6 5 1 2 4	28.31	0.160	9	Rule 7, 5, or 4
3	28.31	0.110	9	5 7 4 3 2 1 6	28.31	0.110	7	Rule 7
4	28.31	0.110	11	5 7 3 4 1 2 6	28.31	0.110	7	Rule 7
5	28.31	0.055	8	5 6 4 2 1 3 7	28.31	0.220	12	Rule 7
6	28.31	0.055	10	7 4 5 6 2 1 3	28.31	0.160	9	Rule 7
7	28.31	0.110	12	5 7 4 2 3 6 1	28.31	0.110	8	Rule 7
8	28.31	0.110	11	5 1 3 6 2 7 4	28.31	0.220	10	Rule 7 or 5
9	28.31	0.055	9	7 2 1 6 4 3 5	28.31	0.160	10	Rule 7
10	28.31	0.055	8	5 7 6 4 3 2 1	28.31	0.160	8	Rule 7 or 4
Mean	*28.31*	*0.088*	*9.7*		*28.31*	*0.160*	*8.9*	

Table 11. Situation of the Two-Factor Combination

Operator Combination		Factor Sequence	Factor Combination
1	$\ldots\bigcap\ldots$	AB	A$\bigcap$B
		BA	B$\bigcap$A
2	$\ldots\bigcup\ldots$	AB	A$\bigcup$B
		BA	B$\bigcup$A

random search. In all the GAs in this paper, we let $C_r = 0.8$ and $M_r = 0.2$. Due to popsize = 200, we get xsize = 160 and msize = 40. The termination criteria for GA is that the algorithm stops when the objective value difference between the worst chromosome and the best one in the current generation is equal to or less than 0.001.

Results of Example 1 by GA combined with different rules

GA was combined with the seven rules in Table 4, respectively, for solving Example 1. Tables 6 and 7 present the results of Example 1 by GA combined with different rules. For each method, 10 tests of computation were performed. The makespan of the optimal schedules is 28.31. It can be seen that from Table 6: (1) Rules 5 and 7 are the best rules that enables the GA to obtain the optimal makespan in every test. (2) Rules 1, 2, and 4 are also good rules, but do not enable GA to obtain the optimal makespan in every test. (3) The other two rules are poor-quality rules that never enable the GA to obtain the optimal makespan. Rule 3 is the worst. An optimal schedule obtained by GA combined with Rule 7 is shown in Figure 5, where $\pi = (8, 7, 4, 2, 1, 6, 9, 3, 10, 5)$, $f(\pi) = C_{\max} = 28.31$.

Table 7 shows that: (1) Rules 3 and 6 are the best rules that enable GA to obtain the optimal F in every test. (2) Rule 2 is also a good rule, but not to enable GA to obtain the optimal F. (3) The other four rules are poor-quality rules that never enable GA to obtain the optimal F. Rule 1 is the worst. Rule 3 is a simple rule, which shows that effective rules for optimized objectives are not always compound rules.

A comparison of Tables 6 and 7 shows that, for the same problem, when the scheduling objective is changed the effective rules are also changed. In order to select the suitable rules, tedious simulating experiments are needed. To overcome this drawback, an ARS method is proposed.

ARS Based on the New Rule Base (ARS1/ARS2)

In our previous work, ARS was studied. The purpose of ARS is to determine the best rule from the rule base S_R (as shown in Table 4) for synthesizing an order sequence into a schedule. GA is still used for this purpose. The basic structure of GA for ARS is the same as described previously (see Figure 4), but the representation of, and the evaluation procedure for a chromosome in GA are changed.

Mixed chromosome and evaluation procedure in ARS

For ARS and later RSE, mixed chromosomes are adopted. Assume that the number of rules in the candidate rule base S_R is R, and natural numbers 1, 2, 3, ..., R are used to denote R rules. A mixed chromosome $P = (P_1, P_2)$ consists of two parts: $P_1 = (\gamma_1, \gamma_2, \gamma_3, \ldots, \gamma_R)$ is a rule sequence, $\gamma_k \in \{1, 2, 3, \ldots, R\}$, $k = 1, 2, 3, \ldots, R$; and $P_2 = (\pi_1, \pi_2, \pi_3, \ldots, \pi_N)$ is an order sequence, $\pi_i \in \{1, 2, 3, \ldots, N\}$, $i = 1, 2, 3, \ldots, N$. For chromosomes in the initial generation, each part of the chromosome is produced randomly. Figure 6 shows a sample mixed chromosome, in which seven rules are available for selection when synthesizing the order sequence into a schedule.

The original evaluation procedure[31] of a chromosome P in ARS can be simplified as the new one shown in Figure 7. The evaluation of a chromsome P is to use the first rule Rule γ_1 in P_1 to synthesize P_2 into a schedule with an objective value, $f(P)$. $f(P)$ can be calculated by the functions 1–6. Every order π_i in P_2 is assigned to the units according to the same Rule γ_1 in P_1. Rule γ_1 will be changed with different chromosomes. Let us call the original ARS as ARS1.

A feasible chromosome is the one that can be synthesized into a schedule by using a rule. In Example 1, there is no CP constraint, so every chromosome is feasible. However, in problems with CP constraints, e.g. Example 3, some chromosomes may be infeasible. In this case, a penalty method is used to handle the CP constraints, which is described in the section of Case Study.

In our new ARS (called ARS2), the new evaluation procedure of a chromosome P is shown in Figure 8. In this procedure, we try all rules in the rule sequence P_1 and select the best one to synthesize P_2 into a schedule. In fact, the mixed chromosomes in ARS2 are not necessary, and P_1 can be omitted. The reason for this is obvious. Through comparison study, we have found that ARS2 performs better than ARS1. ARS2 is capable of obtaining better solutions for large-size problems than ARS1. Why? In ARS2, every current order sequence is synthesized into schedules by all the rules, and the best schedule is used to evaluate the chromosome containing the order sequence. If ARS1 and ARS2 search the same number of chromosomes (with the same popsize) at each iteration, ARS2 explores more schedules than ARS1, though ARS2 is a little more time-consuming. Therefore

Table 12. Calculation of the Number of Factor Combinations

Impact Factors	Operator Combinations	Factor Sequences	Factor Combinations	Equivalent Combinations
2	$2^1 = 2$	$P_2^2 = 2$	$2 \times 2 = 4$	$2\times C_2^2 = 2$
3	$2^2 = 4$	$P_3^3 = 6$	$4 \times 6 = 24$	$C_3^2 + 2\times C_3^3 = 5$
4	$2^3 = 8$	$P_4^4 = 24$	$8 \times 24 = 192$	$C_4^2 + C_4^3 + 2\times C_4^4 = 12$
5	$2^4 = 16$	$P_5^5 = 120$	$16 \times 120 = 1920$	$C_5^2 + C_5^3 + C_5^4 + 2\times C_5^5 = 27$
...	...	...	...	...
n_f	2^{n_f-1}	$P_{n_f}^{n_f}$	$2^{n_f-1}\times P_{n_f}^{n_f}$	$C_{n_f}^2 + C_{n_f}^3 + \cdots + 2\times C_{n_f}^{n_f}$

Table 13. Factor Combinations and Equivalent Combinations of the Three-Factor Problem

Operator	Combination	Factor Sequence	Combination: Factor Combination	Combination: Equivalent Combination	Example: Factor Combination	Example: Equivalent Combination	Example: Rules Concerned
1	…∩…∩…	ABC	A∩B∩C	A∩B∩C	PsT∩CT∩PT	PsT∩CT∩PT	ECT
		ACB	A∩C∩B		PsT∩PT∩CT		
		BAC	B∩A∩C		CT∩PsT∩PT		
		BCA	B∩C∩A		CT∩PT∩PsT		
		CAB	C∩A∩B		PT∩PsT∩CT		
		CBA	C∩B∩A		PT∩CT∩PsT		
2	…∩…∪…	ABC	A∩B∪C	A∩B∪C	PsT∩CT∪PT	PsT∩CT∪PT	EST∪SPT
		BAC	B∩A∪C		CT∩PsT∪PT		
		BCA	B∩C∪A	B∩C∪A	CT∩PT∪PsT	CT∩PT∪PsT	SCPT∪FAU
		CBA	C∩B∪A		PT∩CT∪PsT		
		ACB	A∩C∪B	A∩C∪B	PsT∩PT∪CT	(PsT∩PT)∪CT	SPsPT∪SCT
		CAB	C∩A∪B		PT∩PsT∪CT		
3	…∪…∩…	ABC	A∪(B∩C)	A∪(B∩C)	PsT∪(CT∩PT)	PsT∪(CT∩PT)	FAU∪SCPT
		ACB	A∪(C∩B)		PsT∪(PT∩CT)		
		BAC	B∪(A∩C)	B∪(A∩C)	CT∪(PsT∩PT)	CT∪(PsT∩PT)	SCT∪SPsPT
		BCA	B∪(C∩A)		CT∪(PT∩PsT)		
		CAB	C∪(A∩B)	C∪(A∩B)	PT∪(PsT∩CT)	PT∪(PsT∩CT)	SPT∪EST
		CBA	C∪(B∩A)		PT∪(CT∩PsT)		
4	…∪…∪…	ABC	A∪B∪C	A∪B∪C	PsT∪CT∪PT	PsT∪CT∪PT	FAU∪SCT∪SPT
		ACB	A∪C∪B		PsT∪PT∪CT		
		BAC	B∪A∪C		CT∪PsT∪PT		
		BCA	B∪C∪A		CT∪PT∪PsT		
		CAB	C∪A∪B		PT∪PsT∪CT		
		CBA	C∪B∪A		PT∪CT∪PsT		
Total			*24*	*5*	*24*	*5*	*7*

ARS2 has more chances to find better solutions than ARS1, though ARS2 is more time-consuming.

Results of Example 1 by ARS

Due to the evolution mechanism, GA will automatically select a suitable rule to synthesize an evolved order sequence into a high quality schedule. Table 8 presents the results of minimization of the makespan ($C_{\max}$) for Example 1 by ARS1 and ARS2. ARS1 automatically selected Rule 5 or 7 to synthesize the schedule and achieved the optimal makespan in every test of computation. Table 9 presents the results of minimization of the total flow time (F) for Example 1 by ARS1 and ARS2. ARS1 automatically selected Rule 3 or 6 to synthesize the schedule and achieved the optimum in every test of computation. It can be seen that the effective rules automatically selected by ARS are consistent with the results of the simulation experiments of GA combined with different rules.

From Tables 8 and 9, we can see that both ARS1 and ARS2 find the best solution of Example 1. But ARS2 is able to find the best solution within a smaller number of iterations. As in ARS2 to synthesize an order sequence into a schedule, all the rules, rather than one rule in ARS1, are tried and the best rule selected, it is reasonable that ARS2 is a little more time-consuming. If the problem size increases (e.g. examples in the Case Study), ARS2 is still able to find the similar or better solutions to the problem within a smaller number of iterations.

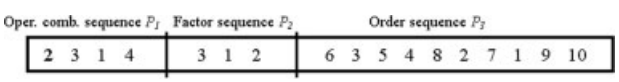

Figure 11. A sample mixed chromosome for ARC.

RSE Based on the New Rule Base (RSE1/RSE2)

In our previous work, RSE was also studied. In ARS1, only one rule is used for assigning all of the orders in an order sequence to the units. If the rule failed, a new chromosome has to be generated. To increase the chance for an order sequence to form a feasible schedule, all rules in the candidate rule base S_R can be tried. That is why a rule sequence is used for assigning an order. The use of a rule

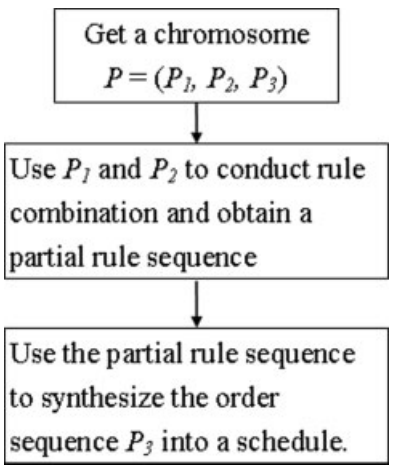

Figure 12. Evaluation procedure in ARC.

Table 14. Rule Combinations and the Corresponding Partial Rule Sequences

Operator Combination		Equivalent Combination	Example		
			Equivalent Combination	Rules Concerned	Rule Sequence
1	...∩...∩...	A∩B∩C	PsT∩CT∩PT	ECT	7 0 0
		A∩B∪C	PsT∩CT∪PT	EST∪SPT	4 3 0
2	...∩...∪...	B∩C∪A	CT∩PT∪PsT	SCPT∪FAU	6 1 0
		A∩C∪B	(PsT∩PT)∪CT	SPsPT∪SCT	5 2 0
		A∪(B∩C)	PsT∪(CT∩PT)	FAU∪SCPT	1 6 0
3	...∪...∩...	B∪(A∩C)	CT∪(PsT∩PT)	SCT∪SPsPT	2 5 0
		C∪(A∩B)	PT∪(PsT∩CT)	SPT∪EST	3 4 0
4	...∪...∪...	A∪B∪C	PsT∪CT∪PT	FAU∪SCT∪SPT	1 2 3

sequence can increase the feasiblity of an order sequence. The aim of RSE is to find out the best rule sequnence to synthesize an order sequence into a schedule. GA is still applied for this porpose. In RSE, chromosome representation is the same as that in ARS, and the basic stucture of GA is the same as before (see Figure 4). The only difference is the evaluation of chromosome P with a rule sequence.

Evaluation procedure in RSE

The original evaluation procedure of a chromosome P by a rule sequence P_1 is shown in Figure 9. If a rule γ_k in P_1 fails to assign an order π_i in P_2, the next rule is tried until the order is assigned successfully. If all rules fail to assign an order to one of the units, then the chromosome is infeasible, and a new chromosome is regenerated. In Example 1, without CP constraints, forbidden assignments do not exist. However, in coming Example 3, with CP constraints, forbidden assignments do exist. For instance, in case of forbidden changeover, the rule shortest changeover time (SCT) may fail to assign an order. The rule sequence P_1 will be changed with different chromosomes. At the end of GA, the best P_1 will be evolved, and an evolved order sequence P_2 is synthesized into a high quality schedule by P_1. Let us call the original RSE as RSE1.

In our new RSE (called RSE2), the new evaluation procedure of a chromosome P is shown in Figure 10. In this procedure, we try all rules in the rule sequence P_1 and select the best feasible rule to assign π_i to a unit. The criterion to judge the rule used in RSE2 depends on the scheduling objective to be optimized. For the minimization of makespan, the completion time of the order is the criterion to judge the rule used. Similarly, the mixed chromosomes in RSE2 are not necessary, i.e., P_1 can be omitted. Through comparison study, we have found that RSE1 performs better that RSE2. Why? In RSE2, trying to use the best rule to assign a local π_i among an order sequence, the schedule from the total order sequence is not certainly of high quality. The objective to be optimized is the performance criteria of the final schedule in which all the orders are contained, but not the performance criteria of a single order.

Results of Example 1 by RSE

Table 10 presents the results of minimization of the makespan ($C_{\max}$) for Example 1 by RSE1 and RSE2. RSE1 finally evolved a rule sequence to synthesize the schedule and achieved the optimum in every test, and Rule 7 or 5 plays the best role in the rule sequence. In RSE2, the final schedule was synthesized by Rule 7, 5, or 4. In most tests, only Rule 7 was selected to synthesize the final schedule. In some tests, two or three rules were used.

ARC Approaches

In ARS and RSE, a full rule sequence is utilized to synthesize an order sequence into a schedule. In fact, for a particular problem, some impact factors are not such significant as the others. As consequence, the related rules are not such important as the others. Therefore, if full rule sequences are always used, the less important rules waste computational resources. That is why we propose ARC approaches. GA is still used for this purpose. The basic structure of GA for the ARC is the same as previous (see Figure 4), but the representation and the evaluation procedure of a chromosome in the GA are changed.

Table 15. The Ways of Schedule Synthesis of a Chromosome in Different ARC Methods

Approach	Description of Schedule Synthesis of a Chromosome $P = (P_1, P_2, P_3)$	Quality
ARC1	Try all rules in the partial rule sequence and select the best one to synthesize the entire order sequence P_3 into a schedule (Each order is assigned by the same rule)	Good
ARC2	Try all rules in the partial rule sequence and select the best one to assign each order π_i in P_3 to a unit (Different order may be assigned by a different rule)	Bad
ARC3	Try one rule in the partial rule sequence to synthesize the entire order sequence P_3 into a schedule (Each order is assigned by the same rule)	Bad
ARC4	Try one rule in the partial rule sequence to assign each order π_i in P_3 to a unit (Different order may be assigned by a different rule)	Good
ARC5	Try *all the equivalent combinations* from P_1 and P_2, and select the best combination to synthesize the entire order sequence P_3 into a schedule	Best

Table 16. Results of Example 1 by ARC (min C_{max})

	ARC1						ARC5			
Test	C_{max}	CPU Time, s	Iterations	o_1 in P_1	P_2	Final Rule Used	C_{max}	CPU Time, s	Iterations	Final Rule Used
1	28.31	0.110	8	1	2 3 1	7	28.31	0.160	9	7
2	28.31	0.055	8	1	1 3 2	7	28.31	0.160	9	7
3	28.31	0.110	9	1	1 2 3	7	28.31	0.110	8	5
4	28.31	0.110	10	1	2 1 3	7	28.31	0.110	7	7
5	28.31	0.11	16	1	1 3 2	7	28.31	0.160	9	5
6	28.31	0.110	12	1	2 1 3	7	28.31	0.160	8	7
7	28.31	0.055	10	1	1 3 2	7	28.31	0.160	9	5
8	28.31	0.055	9	2	1 3 2	5	28.31	0.110	8	5
9	28.31	0.110	11	1	3 1 2	7	28.31	0.160	8	5
10	28.31	0.160	14	2	1 3 2	5	28.31	0.160	9	7
Mean	*28.31*	*0.099*	*10.7*				*28.31*	*0.145*	*8.4*	

Factor combination logic

In schedule synthesis, to increase the feasibility and/or quality of the solution, sometime we must consider two or more factors simultaneously and sometime we need to consider one or another factor. Usually, we rely on logical operators to express the relationships between factors. Factor combination is realized by combination of the operators and the factors. For the factor combination, "AND" and "OR" are used in this paper, and "$\bigcap$" and "$\bigcup$" stand for "AND" and "OR," respectively.

If $n_f = 2$, "A$\bigcap$B" means that we should consider two factors A and B simultaneously, and obviously, we have A$\bigcap$B = B$\bigcap$A. Similarly, "A$\bigcup$B" means that we should consider one of the two factors, A or B, and also we have A$\bigcup$B = B$\bigcup$A. Table 11 shows the situation of the two-factor combination. The situation of three-factor combination is presented in Table 13.

Table 12 shows how to calculate the number of factor combinations. If there are three factors ($n_f = 3$) for combination, then there are $2^2 = 4$ operator sequences, and $P_3^3 = 6$ factor sequences, the total number of factor combinations is $4 \times 6 = 24$. If there are 4 or 5 factors for combination, the total number of factor combinations is 192 or 1920. With the increase of the factors considered, the number of the factor combinations will be huge. Hence, it seems impossible to use every factor combination for schedule synthesis. Fortunately, the number of equivalent combinations is far smaller than the number of the factor combinations. Then, what is an equivalent combination?

Equivalent combination

According to A$\bigcap$B = B$\bigcap$A, if the logical relationships between all the factors (or part of the factors) are "$\bigcap$" in the combinations, then no matter what the sequence of the factors is, these combinations are equivalent, and we can use one of them as an equivalent combination. Similarly, according to A$\bigcup$B = B$\bigcup$A, if the logical relationships between all the factors (or part of the factors) are "$\bigcup$" in the combinations, then no matter what the sequence of the factors is, these two combinations are equivalent, and we can use one of them as an equivalent combination. From Table 12, it is observed that, for a three-factor problem, although there are 24 factor combinations, the number of equivalent combinations is only 5; for a five-factor problem, there are 1920 factor combinations, but the number of equivalent combinations is only 27. Table 13 shows the details of factor combinations and equivalent combinations of the three-factor problem.

Relationship between equivalent combination and rule combination

When the operator $\bigcap$ appears in an equivalent combination, a compound rule is needed. For example, if the CT and the PT of order j are considered simultaneously, Rule 6 (SCPT) is the right rule; if three factors are considered at the same time, Rule 7 (ECT) is the right rule. If the operator $\bigcup$ appears, two or more rules may be needed. For example, if the CT or the PT is considered, Rule 2 or 3 should be used. From Table 13, an equivalent combination may be concerned

Table 17. Differences Between Examples 2, 3 and 4

N, M	Example 2 $N = 50$, $M = 4$	Example 3 $N = 50$, $M = 4$	Example 4 $N = 100$, $M = 8$
c_{ij} and p_{ju}	Without CP constraints, no c_{ij} in the changeover time matrix is null and no p_{ju} in process time matrix is null	With CP constraints, some of c_{ij} in the changeover time matrix are null and some of p_{ju} in process time matrix are null	Without CP constraints, no c_{ij} in the changeover time matrix is null and no p_{ju} in process time matrix is null. c_{ij} and p_{ju} are generated randomly: $c_{ij} \in (0.10, 2.00)$, $p_{ju} \in (5.0, 20.00)$.
d_j	With a finite value	With a finite value	d_j is generated by random, $d_j \in [20, 80]$
or_j	$or_j = 0$	With a finite value	or_j is generated by random, $or_j \in [0, 6]$
ur_u	$ur_u = 0$	With a finite value	ur_u is generated by random, $ur_u \in [0, 3]$

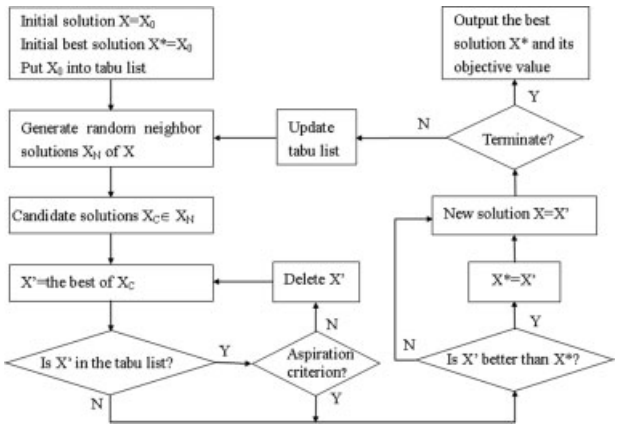

Figure 13. Flow chart of tabu search.

with one rule, two rules, or three rules. If all the operators between the factors are "$\cap$", the factor combinations can be changed into an equivalent combination that corresponds to one compound rule. If "$\cup$" exists in the factor combination, its equivalent combination corresponds to two rules or three rules. For Example, the equivalent combination PsT$\cup$CT$\cup$PT corresponds to a three-rule sequence FAU$\cup$SCT$\cup$SPT (1 2 3). Generally, a three-rule sequence can be used to correspond to an equivalent combination. If the number of rules is less than three in the rule sequence, then "null" rule (0) is used to supplement the rule sequence. For example, PsT$\cap$CT$\cap$PT corresponds to EST$\cup$null$\cup$null (7 0 0). Contrasted to a full rule sequence in ARS or RSE that consists of all the rules in the rule base, a three-rule sequence is a partial rule sequence.

What kind of logic should be applied when using a partial rule sequence to synthesize an order sequence into a schedule? If a partial rule sequence involves one rule, just use the rule for schedule synthesis as before. If a partial rule sequence involves two rules or three rules, the following logic can be utilized to evaluate a order sequence: (a) use any one rule for schedule synthesis; (b) select the best rule for schedule synthesis; (c) try one rule first, if it fails, then try another rule. Therefore, the operator "$\cup$" in the rule sequence has the meaning of "OR" or "THEN." In coding of the algorithm, "OR" or "THEN" can be selected. Up to now, the purpose of factor combination is evident. Through factor combination, a partial rule sequence can be obtained to synthesize an order sequence into a schedule.

Representation and evaluation of a mixed chromosome in ARC

Assume that the number of the factors is n_f, and natural numbers 1, 2, 3, ..., n_f are used to denote the n_f factors. A mixed chromosome $P = (P_1, P_2, P_3)$ consists of three parts: $P_1 = (o_1, o_2, o_3, \ldots, o_{2^{n_f-1}})$ is a operator combination sequence, $o_k \in \{1, 2, 3, \ldots, 2^{n_f-1}\}$, $k = 1, 2, 3, \ldots, 2^{n_f-1}$; $P_2 = (\varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_{n_f})$ is a factor sequence, $\varphi_j \in \{1, 2, 3, \ldots, n_f\}$, $j = 1, 2, 3, \ldots, n_f$; and $P_3 = (\pi_1, \pi_2, \pi_3, \ldots, \pi_N)$ is an order sequence, $\pi_i \in \{1, 2, 3, \ldots, N\}$, $i = 1, 2, 3, \ldots, N$. For the chromosomes in initial generation, each part of the chromosome is produced randomly. Figure 11 shows a sample mixed chromosome in ARC, in which there are three ($n_f = 3$) factors considered, and $2^{n_f-1} = 4$ operator combinations (see Table 12).

The order sequence P_3 in a mixed chromosome P would be synthesized into a schedule according to the operator combination sequence P_1 and the factor sequence P_2 with two steps (see Figure 12):

Step one: Utilize the operator combination sequence P_1 and the factor sequence P_2 to conduct rule combination based on the principle shown in Table 13, and obtain a rule sequence (see Table 14).

Let us take the sample chromosome in Figure 11 as an example. For the first digit "2" in the operator combination sequence P_1, its corresponding operator combination is "...$\cap$...$\cup$...."; the factor sequence P_2 is "3 1 2" (that is "CAB" in Table 13). According to Table 13, we get equivalent combination "A$\cap$C$\cup$B." In Example 1, the equivalent combination is "(PsT$\cap$PT)$\cup$CT," and hence the rules concerned is "SPsPT$\cup$SCT." Eventually, we get the partial rule sequence "5 2 0," which is the partial rule sequence to be used in Step two for schedule synthesis. For the other three digits "3, 1, 4" in the operator combination sequence, their equivalent combinations and corresponding partial rule sequences can be achieved in a similar way.

Step two: Utilize the partial rule sequence to synthesize the order sequence P_3 into a schedule. In the previous sections, we discussed how to select one rule or use a (full) rule sequence to synthesize an order sequence into a schedule. Here the partial rule sequence involves one rule (at least) to three rules (at most). Table 15 shows the ways of schedule synthesis of a chromosome $P = (P_1, P_2, P_3)$ in different ARC approaches (ARC1–5). In ARC1–4, only the first digit of the operator combination sequence P_1, together with the factor sequence P_2, is used for rule combination in Step one, so that only one of the partial rule sequences in Table 14 is used for schedule synthesis of P_3 in Step two. In contrast, in

Table 18. Results of Example 2 by MILPs (min C_{max})

	Original Model (Big M = 1000)				Improved Model					
	100,000 Iterations		10,000,000 Iterations		100,000 Iterations				10,000,000 Iterations	
N	C_{max}	CPU, s	C_{max}	CPU, s	n	Big M	C_{max}	CPU, s	C_{max}	CPU, s
8	14.00	16.60	14.00	16.60	4	136.6	14.00	1.15	14.00	1.15
10	19.10	26.67	18.35	6254.36	5	251	17.70	26.69	17.70	5741.08
15	28.00	44.44	26.60	6189.41	7	298.5	25.40	45.57	25.40	5675.48
20	38.85	72.45	35.50	8831.61	9	413	32.35	70.71	32.35	7816.14
50	106.40	600.38	103.85	60,240.51	24	499	98.84	593.91	98.84	59,881.41

Table 19. Results of Example 2 by GAs (min C_{max})

	GAs (Best Test)				
	GA_R7		ARS2		
N	C_{max}	CPU, s	C_{max}	CPU, s	Rule
8	14.00	<0.055	14.00	<0.055	Rule 7
10	17.35	0.055	17.35	0.16	Rule 7
15	22.19	0.11	22.05	0.49	Rule 7
20	27.89	0.16	27.80	1.04	Rule 6
50	74.30	0.77	70.30	2.69	Rule 6

ARC5, all these four digits of an operator combination sequence P_1 are used for rule combination in Step one, and thus, four of the partial rule sequences in Table 14 are tried in Step two, and the best partial rule sequence is selected for schedule synthesis of P_3. Through computational experiments with different examples, ARC2 and ARC3 performed badly. The reasons for this are the same as in RSE2 and ARS1. Therefore, the results by ARC2, ARC3, and ARC4 are not presented in this paper.

ARC1, compared with ARS2, and ARC4, compared with RSE1, use a smaller number of rules for schedule synthesis for every chromosome in the course of the computation. In ARC5, to evaluate a chromosome, four equivalent combinations, and hence four corresponding partial rule sequences are tried. Consequently, more CPU time is needed.

Results of Example 1 by ARC

Table 16 shows the results of minimization of the makespan (C_{max}) for Example 1 by ARC1 and ARC5. In both ARC1 and ARC5, Rule 5 or 7 is finally used for schedule synthesis, and the optimal makespan is achieved in every test of computation. By decoding the final evolved chromosome corresponding to the best schedule obtained in ARC, we can determine the important impact factors, the concerned factor combinations, and rules.

Case Study

In addition to Example 1, the following three examples are used for case study. Examples 2 and 3 were widely used in the literature.[10–13] The authors often solved small-size problems by using MILP. In this paper, the problem is enlarged, and it is very difficult for MILP to obtain an acceptable solution to these large-size problems. Example 4 is a randomly generated problem, a very large general problem. The data for Examples 2–4 were presented in the Supporting Information (see Supplementary Material for this article). Table 17 shows the differences between the three examples.

Our approaches can cope with problems with forbidden changeovers, forbidden processes, order/unit release times (like Example 3), and problems with randomly generated order/unit release times (like Example 4). In Examples 2 and 4, there is no CP constraint. But in Example 3, there are CP constraints. To increase feasibility of the solutions explored and reduce the search time, a penalty method can be used to change the problem with CP constraints into that without CP constraint.

To compare the performance of MILPs and GAs, these two kinds of approaches are used to solve different size problems in Example 2. For the large-size problems, e.g. 50-order problems in Examples 2 and 3 and 100-order problem in Example 4, since it is obvious that MILPs cannot provide good enough solutions for comparison, the results by MILPs are not very important.

With an increase in problem size, the solution quality of GA degrades rapidly. That is to say, GA prematurely converges. Basic GA is one of the meta-heuristic methods, but it may not be the best one for combinatorial optimization. In solving large-size combinatorial problems including scheduling problems, another kind of meta-heuristic method, TS, shows persistent search ability, though with lower convergence speed than GA. In practice, long time TS is often used to search the best solution to large combinatorial problems. That is why we provide the results by long time TS for the large-size problems in these three examples.

TS is a neighborhood search method, like SA. TS uses memory (called tabu list) to keep track of solutions already visited, so it considers historical information during the search process to avoid unnecessary revisit. Therefore, TS searches the neighborhood of a solution fully but cleverly. But TS that uses only one solution for the coming iteration can easily miss some promising areas of the search space, and a large set of parallel solutions does not exchange information. Aspiration criteria adopted in TS introduce the algorithm to search some secondary solution space. Lin and Miler[37] presented a TS for continuous function optimization in chemical process. The TS procedure for combinatorial optimization is similar to that for the continuous function optimization. The TS procedure consists of several steps which are depicted in Figure 13. The components of TS are summarized as follows:

Table 20. Results of Example 2 by GAs Combined with Different Rules and TS (min C_{max})

Method	Best C_{max}	Mean C_{max}	Mean Dev. from Best	Mean CPU Time, s	Mean Iterations	Tests	Rule Combined
GA_R1 (FAU)	80.34	83.02	18.51	0.80	48.5	10	Rule 1
GA_R2 (SCT)	82.00	84.96	21.29	0.81	52.7	10	Rule 2
GA_R3 (SPT)	96.20	96.78	38.16	0.51	34.3	10	Rule 3
GA_R4 (EST)	81.25	84.41	20.50	0.67	41.4	10	Rule 4
GA_R5 (SPsPT)	74.60	75.48	7.75	0.77	47.2	10	Rule 5
GA_R6 (SCPT)	71.29	72.06	2.86	0.61	37.6	10	Rule 6
GA_R7 (ECT)	74.30	75.61	7.94	0.70	44.2	10	Rule 7
TS_R6 (SCPT)	70.05	70.45	0.57	41.5	10,000	10	Rule 6

Table 21. Results of Example 2 by Different Methods (min C_{max})

Method	Best C_{max}	Mean C_{max}	Mean Dev. from Best	Mean CPU Time, s	Mean Iterations	Tests	Rule or Rule Sequence
ARS2	70.30	71.59	2.20	2.47	43.5	10	6
RSE1	70.70	71.51	2.09	1.05	55.8	10	6 3 7 4 5 2 1
ARC1	70.70	73.01	4.23	1.18	50.8	10	6
ARC5	70.13	71.18	1.61	3.65	56.4	10	6

Encoding: Encoding is the representation of a solution, the same as that in GA.

Initial solution and evaluation: TS begins with a single initial solution, this solution is evaluated by the same procedure as shown in Figure 13. Other solutions during the TS process are evaluated by the same procedure.

Objective function: Through objective function, every solution corresponds to an objective value.

Move operator: Like the mutation of GA, it changes the permutation of a solution and results in another solution.

Neighbor solutions: Through move operator, a set of neighbor solutions can be obtained from one solution.

Candidate solutions: Candidate solutions are a subset of the neighbor solutions.

Tabu list: The solutions that have just been searched are put into the tabu list, and endued with a certain or dynamic tenure. At each iteration the tenure will be subtracted by 1. If the tenure is not equal to zero, the solution will not be searched, that is to say, it will not be used to generate neighbor solutions. The tabu list length depends on the problem size.

Aspiration criterion: Aspiration criterion is used to determine whether a candidate solution is accepted or rejected to generate neighbor solutions. If all candidate solutions are in the tabu list, the best candidate is accepted. If a solution is the best so far, that means there are not any solutions better than it, it is accepted without checking the tabu list.

Termination criterion: If the termination criterion is satisfied, then output the best solution. We use a number of iterations, 10,000, as termination criterion in this paper.

Example 2

In this section, we first solve Example 2 using the MILP model developed by Hui and Gupta[12] and the modified model and then solved them using the newly proposed approaches. The MILP procedure for solving SMSP given by Hui and Gupta[12] is as follows: establish the MILP model first, including objective function and constraints, and then the model is formulated by GAMS and solved by OSL solver. When using GAMS and OSL for solving the MILP model, iterations is limited to 100,000 or 10,000,000 for the computation. For the small-size problems, if optimal solution is obtained, the computational process will stop with a small number of iterations. Table 18 shows the results of Example 2 by MILPs, and Table 19 shows the results by our proposed GAs.

Five observations can be made from Tables 18 and 19. First, as the problem size increases linearly, from 8-order/4-unit to 50-order/4-unit, the computational time of MILP model increases exponentially, from 2 to about 600 s (with a termination criterion of 100,000 iterations), or from 2 to about 60,000 s (about 17 h) (with a termination criterion of 10,000,000 iterations). Second, although the solutions by the improved MILP model are somewhat better, the CPU times do not reduce correspondingly. Third, although the iterations increase by 100 times, the objective values by the original model decrease very little, and the objective values by the improved model have no change. Fourth, the proposed GAs obtain much better solutions for large-size problems within much shorter search time than MILPs. Finally, ARS2 performs better in solving large-size problems than GA_R7 which stands for GA combined with Rule 7. Comparing the results between MILPs in Table 18 and GAs in Table 19 shows that GAs take great advantage over MILPs, especially in solving large-size problems.

To compare the performance of the seven rules in GA, Example 2 with 50 orders over 4 units was solved by GAs combined with different rules. For each method, 10 tests were conducted. The best makespan of Example 2 obtained by long time TS combined with Rule 6 (TS_R6) up to now is 70.05. Hence the deviations from best for Example 2 are calculated with respect to 70.05. Table 20 presents the

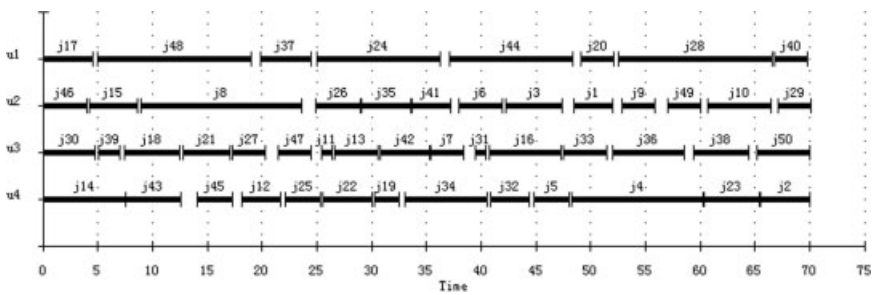

Figure 14. Gantt chart of the schedule for Example 2 by ARC5 (C_{max} = 70.13).

Table 22. Results of Example 3 by GAs Combined with Different Rules and TS (min C_{max})

Method	Best C_{max}	Mean C_{max}	Mean Dev. from Best	Mean CPU Time, s	Mean Iterations	Tests	Rule Combined
GA_R6 (SCPT)	99.30	102.16	7.11	0.88	58.7	10	Rule 6
GA_R7 (ECT)	97.02	100.99	5.54	0.80	51.8	10	Rule 7
TS_R7 (ECT)	95.69	96.74	1.09	70	10,000	10	Rule 7

results. It can be seen that: (1) Rule 6 enables the GA to obtain the best makespan 71.29 among the seven rules; (2) The mean Dev. from best of Rule 6 is the smallest; (3) The mean CPU times are not so different between methods. Therefore Rule 6 is the best rule for Example 2. Compared with Example 1, as the problem size of Example 2 is enlarged, the effective rule for Example 2 changes from Rule 1 to Rule 6.

Table 21 summarizes the results of minimizing the makespan of Example 2 by ARS2, RSE1, ARC1, and ARC5. Rule 6 shows the best performance in solving Example 2 with 50 orders over 4 units. ARC5 achieved the best makespan 70.13 among the four methods. Figure 14 is a Gantt chart of the schedule for Example 2 by ARC5. Compared with the long time TS (TS_R6 in Table 20), ARC5 had a higher mean deviation from best, but ARC5 had much shorter computational time.

Example 3

Example 3 is an SMSP with CP constraints. In GA for solving SMSP with CP constraints, infeasible chromosomes will surely be generated in these steps: "Initial generation," "Crossover," "Mutation." In this case, new feasible chromosomes should be regenerated to replace the infeasible ones. It will take a lot of CPU time for GA to generate feasible chromosomes.

To increase computing speed of GA for problems with CP constraints, a penalty method has been adopted,[31] which gives the forbidden changeover (from one order to another order) or forbidden process (meaning that an order is forbidden to process over a unit) a large penalty numerical value: a great CT or PT. For instance, change the null values in CT matrix and PT matrix into 100.0. Therefore, every arbitrary chromosome became a legal or feasible one. As a result, CPU time is greatly reduced. Moreover, GA obtained similar solutions as before.

To compare the performance of the seven rules in GA, Example 3 with 50 orders over 4 units was also solved by GAs combined with different rules. For each method, 10 tests were conducted. The best makespan of Example 3 obtained by long time TS combined with Rule 7 (TS_R7) up to now is 95.69. Hence the deviations from best for Example 3 are calculated with respect to 95.69. Table 22 presents the results by GA_R6, GA_R7, and TS_R7. Since Rules 1–5 demonstrated poor performance in minimizing the makespan, the results by from GA_R1 to GA_R5 are not presented here. It is obvious that Rule 7 is the best rule for minimization of the makespan of Example 3 with 50 orders over 4 units.

Table 23 summarizes the results of minimizing the makespan of Example 3 by ARS2, RSE1, ARC1, and ARC5. Rule 7 shows the best performance in solving Example 3 with 50 orders over 4 units. ARC5 achieved the best makespan 96.60 among the four methods. Figure 15 is a Gantt chart of the schedule for Example 3 by ARC5. Compared with the long time TS (TS_R7 in Table 22), ARC5 had higher mean deviation from best, but ARC5 had much shorter computational time.

For the same size problems (50 orders over 4 units), the effective rule for Example 3 is different from that for Example 2. The main difference between these two examples is that Example 3 has CP constraints, but Example 2 has no CP constraint.

Example 4

Example 4 is a general large-size problem. Due to the combinatorial explosiveness of the problem, it is very difficult for MILP to solve this example to its optimality within acceptable CPU time. The best makespan of Example 4 obtained by long time TS combined with Rule 6 (TS_R6) up to now is 94.74. Hence deviations from best for Example 4 are calculated with respect to 94.74. Table 24 presents the results of minimization of the makespan for Example 4 by GA_R5, GA_R6, GA_R7, and TS_R6. Through the simulation experiments, it is easy to find that Rule 6 is the best rule for the minimization of the makespan of Example 4.

Table 25 summarizes the results of minimizing the makespan of Example 4 by ARS2, RSE1, ARC1, and ARC5. Rule 6 shows the best performance in solving Example 4. ARC5 achieved the best makespan 96.27 among the four methods. Figure 16 is a Gantt chart of the schedule for Example 4 by ARC5. Compared with the long time TS (TS_R6 in Table 24), ARC5 had higher mean deviation from best, but ARC5 had much shorter computational time.

Table 23. Results of Example 3 by Different Methods (min C_{max})

Method	Best C_{max}	Mean C_{max}	Mean Dev. from Best	Mean CPU Time, s	Mean Iterations	Tests	Rule or Rule Sequence
ARS2	97.30	100.37	4.89	3.66	62.7	10	7
RSE1	98.05	101.06	5.61	1.27	66	10	7 2 1 4 6 3 5
ARC1	98.15	101.11	5.66	1.36	61.7	10	7
ARC5	96.60	99.17	3.64	3.66	55.3	10	7

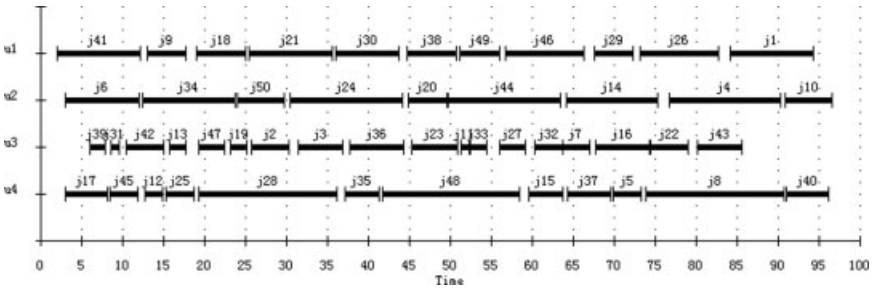

Figure 15. Gantt chart of the schedule for Example 3 by ARC5 (C_{max} = 96.60).

Table 24. Results of Example 4 by GAs Combined with Different Rules and TS (min C_{max})

Method	Best C_{max}	Mean C_{max}	Mean Dev. from Best	Mean CPU Time, s	Mean Iterations	Tests	Rule Combined
GA_R5 (SPsPT)	105.52	106.98	12.91	2.41	50.7	10	Rule 5
GA_R6 (SCPT)	97.30	99.36	4.87	2.09	46.5	10	Rule 6
GA_R7 (ECT)	103.81	105.28	11.13	2.20	46.2	10	Rule 7
TS_R6	94.74	95.76	1.07	184.5	10,000	10	Rule 6

Table 25. Results of Example 4 by Different Methods (min C_{max})

Method	Best C_{max}	Mean C_{max}	Mean Dev. from Best	Mean CPU Time, s	Mean Iterations	Tests	Rule or Rule Sequence
ARS2	96.94	98.61	4.09	10.26	53.6	10	6
RSE1	97.39	101.36	6.99	3.51	70	10	6 5 2 7 3 4 1
ARC1	99.53	102.94	8.65	3.82	54	10	6
ARC5	96.27	97.99	3.43	13.89	65.6	10	6

Discussion

Our case study shows that: (1) With the change of scheduling objective, the effective rules would change; (2) For the same scheduling objective, with the change of prevailing shop or plant conditions, such as problem size, constrains, PT, CT, and so on, the effective rules would change. Therefore, traditionally, a large number of simulation experiments are needed to select suitable rule(s) for diverse scheduling objectives. Moreover, if the conditions changed, new simulation experiments should be carried out again. By the novel rule evolutionary approach, every time when solving the problem, the algorithm will automatically select the suitable heuristic rules. Our task is to encode the rules into the algorithms, making them more "clever" or intelligent.

Representation simplification

In ARS2 and RSE2, a rule sequence in a chromosome is not necessary, because every rule in the candidate rule base is tested in synthesizing an order sequence into a schedule. Similarly, in ARC5, the operator combination sequence in a chromosome is not necessary, because every operator sequence is tested to handle the order sequences. Thus, the representation of a solution in these methods can be simplified. Consequently, computational resource can be saved and computational speed can be increased without any adverse effect on solution quality. The simplified representation makes it easy for ARS and rule combination to be used in other meta-heuristics, such as TS and hybrid methods.

Compound objectives

As stated previously, the makespan and the total tardiness, or, the total flow time and the total tardiness, can be conflicting objectives. If the due dates of the orders are loose, it is easy to find a schedule with zero total tardiness, however, the makespan or the total flow time of the schedule may be long. On the other hand, a schedule with short makespan or total flow time may have orders finished with violation of due dates (with tardy orders). To consider both objectives at the same time, compound objectives *TC* and *TF* can be minimized by our proposed approaches.[38]

Table 26 summarizes the results of C_{max} and *F* minimization by ARC5. Table 27 summarizes the results of *TC* and *TF* minimization by ARC5. By comparing Table 26 with Table 27, it can be observed that: (1) for the same example, the effective rule for C_{max} is also effective for *TC*, and the effective rule for *F* also effective for *TF*; (2) when minimizing C_{max} or *F*, the final optimized schedule often has a large total tardiness, but with the compound objective (e.g. *TC*) minimized, both simple objectives included (e.g. C_{max} and *T*) are optimized. In Example 1, when minimizing *TC*, Rule 5 or 7 is finally used, and the best *TC* = 28.79 is achieved. At the same time, zero tardiness and short makespan (28.79) are obtained. For *TF* minimization in Example 1, the suitable

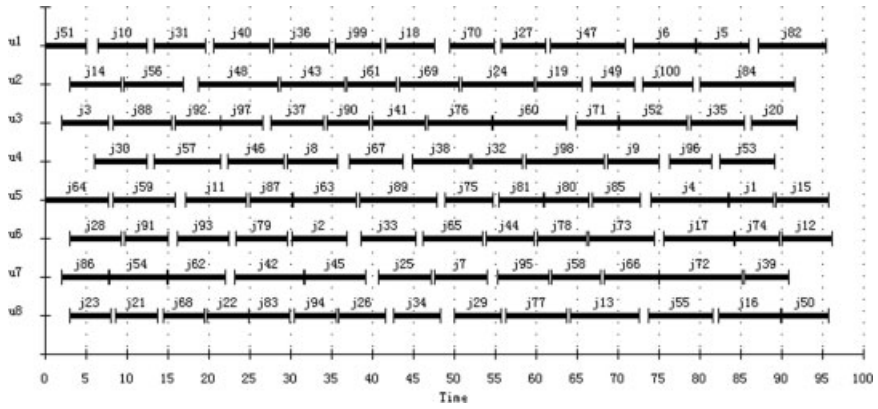

Figure 16. Gantt chart of the schedule for Example 4 by ARC5 (C_{max} = 96.27).

Table 26. Summarized Results of C_{max} and F Minimization by ARC5

	min C_{max} (10 Tests Conducted)				min F (10 Tests Conducted)			
	Schedule with Best C_{max}				Schedule with Best F			
Example	C_{max}	T	Mean C_{max}	Rule Used	F	T	Mean F	Rule Used
Example 1	28.31	6.31	28.31	Rule 5 or 7	73.39	14.62	73.39	Rule 3 or 6
Example 2	70.13	171.50	71.18	Rule 6	271.78	228.43	273.46	Rule 6
Example 3	96.60	573.8	99.17	Rule 7	353.58	562.71	356.01	Rule 6
Example 4	96.27	985.25	97.99	Rule 6	712.69	1273.86	716.38	Rule 6

Table 27. Summarized Results of *TC* and *TF* Minimization by ARC5

	min *TC* (10 Tests Conducted)					min *TF* (10 Tests Conducted)				
	Schedule with Best *TC*					Schedule with Best *TF*				
Example	*TC*	*T*	C_{max}	Mean *TC*	Rule Used	*TF*	*T*	*F*	Mean *TF*	Rule Used
Example 1	28.79	0.00	28.79	28.79	Rule 5 or 7	73.49	0.00	73.49	73.49	Rule 3 or 6
Example 2	81.67	7.50	74.17	86.50	Rule 6	288.95	7.52	281.43	295.01	Rule 6
Example 3	449.66	348.92	100.74	497.10	Rule 6 or 7	707.25	342.64	364.61	745.03	Rule 6 or 7
Example 4	470.08	370.59	99.49	532.69	Rule 6	1084.76	348.85	735.91	1140.50	Rule 6

rule shifts to Rule 3 or 6, and the best TF = 73.49 is achieved. At the same time, zero tardiness and short total flow time (73.49) are obtained.

Conclusions

In solving large-size scheduling problems, it is still very difficult for MILP method to obtain the optima within acceptable computational time. Comparatively, meta-heuristic methods can easily obtain "good enough" solutions to large-size problems within reasonable computational time, though they cannot prove optimality. In meta-heuristic methods, heuristic rules play a very important role in reducing the search space. In our previous work, ARS and RSE methods have been proposed which work with a candidate rule base. This work proposes a novel method of how to construct a comprehensive, but not very large rule base that does not let possible useful rules lie out of our consideration based on the impact factors analysis. Based on the new rule base, new ARS and RSE methods (ARS2 and RSE2) are proposed and investigated. ARS2 shows better performance than the original one.

In ARS and RSE methods, a full rule sequence is used to synthesize an order sequence into a schedule. In the new ARC approach, through impact factor combination, a partial rule sequence is intentionally formed and used for schedule synthesis of an order sequence. The adoption of the partial rule sequences save computational sources and increase the search ability of the algorithms. Through case study, it has been found that ARC approach (ARC5) has almost the same search ability as long time TS to find near-optimal solutions to the large-size problems. Moreover, ARC5 has a 10 times higher convergence speed than long time TS.

Acknowledgments

The authors gratefully acknowledge financial support from Hong Kong RGC grant (No. 614005) and DAG05/06.EG23.

Notation

Abbreviations

ARC = automatic rule combination by GA
ARS = automatic rule selection by GA
CP constraints = changeover constraints and processing constraints
ECT = earliest completion time
EDD = earliest due date
EST = earliest start time
FAU = first available unit
GA = genetic algorithm
GA_Rn = GA combined with Rule n
LCT = latest completion time
MILP = mixed-integer linear programming
MINLP = mixed-integer nonlinear programming
MMSP = multistage multiproduct scheduling problem
MP = mathematical programming
PMX = partial-mapped crossover
RSE = rule sequence evolution by GA
SA = simulated annealing
SMSP = single-stage multiproduct scheduling problem
SCPT = shortest changeover + process time
SCT = shortest changeover time
SPsPT = shortest possible start time + process time
SPT = shortest process time
TS = tabu search
TSP = traveling salesman problem
TS_Rn = TS combined with Rule n

Indices

i, j = different customer orders
u = processing units

Sets

O = a set of orders (with N orders)
S_R = a set of scheduling rules (with R rules)
U = a set of units (with M units)

Parameters

c_{ij} = changeover time when order i changeover to order j (not unit dependant)
CPU, CPU time = computational time
C_r = crossover rate
Dev. from best (%) = deviation from the best objective value
d_j = due date: the committed shipping or completion date of order j (the date the order is promised to the customer)
M = the number of units
msize = the number of the chromosomes that are selected to mutate
M_r = mutation rate
N = the number of orders
n_c = the number of compound rules
n_f = the number of impact factors
n_s = the number of simple rules
or_j = order release time: the earliest time at which order j can start its processing
p_{ju} = order process time
popsize = the number of chromosomes in the initial generation
ur_u = unit release time: the represents the time at which unit u can get ready
xsize = the number of the chromosomes that are selected to crossover
α, β = weight coefficient in compound scheduling objectives

Variables and functions

C_j = completion time of order j
C_{max} = makespan
E = total earliness
E_j = earliness of order j
F = total flow time
$f(P)$ = objective value of P
$f(\pi)$ = objective value of π
P = a mixed chromosome of GA for rule selection or rule evolution: $P = (P_1, P_2)$, $P_1 = (\gamma_1, \gamma_2, \gamma_3, \ldots, \gamma_R)$ is a rule sequence, $P_2 = (\pi_1, \pi_2, \pi_3, \ldots, \pi_N)$ is an order sequence; a mixed chromosome of GA for rule combination: $P = (P_1, P_2, P_3)$, $P_1 = (o_1, o_2, o_3, \ldots, o_{2^{n_f}-1})$ is a operator combination sequence, $P_2 = (\varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_{n_f})$ is a factor sequence, $P_3 = (\pi_1, \pi_2, \pi_3, \ldots, \pi_N)$ is an order sequence
T = total tardiness
TC = compound objective of makespan and total tardiness
TF = compound objective of total flow time and total tardiness
T_j = tardiness of order j
X = a solution in tabu search
π = a permutation $(\pi_1, \pi_2, \ldots, \pi_N)$ in set O

Literature Cited

1. Reklaitis GV. Review of scheduling of process operations. *AIChE Symp Ser*. 1982;78:119–133.
2. Ku HM, Rajagopalan D, Karimi I. Scheduling in batch process. *Chem Eng Prog*. 1987;83:25–34.
3. Reklaitis GV, Sunol AK, Rippin DWT, Hortacsu O. *Batch Processing Systems Engineering*. Berlin: Springer, 1996.
4. Floudas CA, Lin X. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Comput Chem Eng*. 2004;28:2109–2129.
5. Kondili E, Pantelides CC, Sargent RWH. A general algorithm for short-term scheduling of batch operations, Part 1: MILP formulation. *Comput Chem Eng*. 1993;17:211–227.
6. Kallrath J. Planning and scheduling in the process industry. In: Günther HO, Van Beek P, editors. *Advanced Planning and Scheduling Solutions in Process Industry*. Berlin: Springer, 2003:11–42.
7. Pantelides CC. Unified frameworks for optimal process planning and scheduling. In: Rippin DWT, Hale JC, Davis J, editors. *Proceedings of the Second International Conference on Foundations of Computer-Aided Process Operations*. Colorado: Crested Butte, 1993:253–274.
8. Pinto JM, Grossmann IE. A continuous time mixed integer linear programming model for short term scheduling of multistage batch plants. *Ind Eng Chem Res*. 1995;34:3037–3051.
9. Pinto JM, Grossmann IE. A continuous time MILP model for short term scheduling of batch plants with pre-ordering constraints. *Comput Chem Eng*. 1996;20:1197–1202.
10. Cerda J, Henning P, Grossmann IE. A mixed integer linear programming model for short-term scheduling of single-stage multiproduct batch plants with parallel lines. *Ind Eng Chem Res*. 1997;36:1695–1707.
11. Karimi IA, McDonald CM. Planning and scheduling of parallel semi-continuous process. II. Short-term scheduling. *Ind Eng Chem Res*. 1997;36:2701–2714.
12. Hui CW, Gupta A. A bi-index continuous time MILP model for Short-term scheduling of single-stage multi-product batch plants with parallel line. *Ind Eng Chem Res*. 2001;40:5960–5967.
13. Chen C, Liu C, Feng X, Shao H. Optimal short-term scheduling of multiproduct single-stage batch plants with parallel lines. *Ind Eng Chem Res*. 2002;41:1249–1260.
14. Pinto JM, Grossmann IE. Assignment and sequencing models for the scheduling of process systems. *Ann Oper Res*. 1998;81:433–466.
15. Sundaramoorthy A, Karimi IA. A simpler better slotbased continuous-time formulation for short-term scheduling in multiproduct batch plants. *Chem Eng Sci*. 2005;60:2679–2702.
16. Castro PM, Grossmann IE. An efficient MILP model for the short-term scheduling of single stage batch plants. *Comput. Chem Eng*. 2006;30:1003–1018.
17. Castro PM, Barbosa-Povoa AP, Matos HA, Novais AQ. Simple continuous-time formulation for short-term scheduling of batch and continuous processes. *Ind Eng Chem Res*. 2004;43:105–118.

18. Jain V, Grossmann IE. Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS J Comput*. 2001;13:258–276.
19. Maravelias CT, Grossmann IE. A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. *Comput Chem Eng*. 2004;28:1921–1949.
20. Castro PM, Grossmann IE, Novais AQ. Two new continuous-time models for the scheduling of multistage batch plants with sequence dependent changeovers. *Ind Eng Chem Res*. 2006;45:6210–6226.
21. Castro PM, Grossmann IE. New continuous-time MILP model for the short-term scheduling of multistage batch plants. *Ind Eng Chem Res*. 2005;44:9175–9190.
22. Harjunkoski I, Grossmann IE. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Comput Chem Eng*. 2002;26:1533–1552.
23. Mendez CA, Cerda J, Grossmann IE, Harjunkoski I, Fahl M. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Comput Chem Eng*. 2006;30:913–946.
24. Cheng TCE, Sin CCS. A state-of-the-art review of parallel-machine scheduling research. *Eur J Oper Res*. 1990;47:271–292.
25. Park Y, Kim SY, Lee YH. Scheduling jobs on parallel machines applying neural network and heuristic rules. *Comput Ind Eng*. 2000;38:189–202.
26. Mokashi SD, Kokossis AC. Maximum order tree algorithm for optimal scheduling of product distribution lines. *AIChE J*. 2002;48:287–301.
27. He Y, Hui CW. Dynamic rule-based genetic algorithm for large-size single-stage batch scheduling. In: Marquardt W, Pantelides CC, editors. *Proceedings of 16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering*, Garmisch-Partenkirchen, Germany, 2006:1911–1916.
28. Tanev IT, Uozumi T, Morotome Y. Hybrid evolutionary algorithm-based real-world flexible job shop scheduling problem: application service provider approach. *Appl Soft Comput*. 2004;5:87–100.
29. Nowicki E, Smutnicki C. A fast tabu search algorithm for the permutation flow-shop problem. *Eur J Oper Res*. 1996;91:160–175.
30. Grabowsk J, Wodecki M. A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. *Comput Oper Res*. 2004;31:1891–1909.
31. He Y, Hui CW. Rule-evolutionary approach for single-stage multi-product scheduling with parallel units. *Ind Eng Chem Res*. 2006; 45:4679–4692.
32. Pinedo M. *Scheduling Theory Algorithms and Systems*. New Jersey: Prentice Hall/Englewood Cliffs, 1995.
33. He Y, Hui CW. Genetic algorithm for large-size multi-stage batch plant scheduling. *Chem Eng Sci*. 2007;62:1504–1527.
34. Ku HM, Karimi IA. An evaluation of simulated annealing for batch process scheduling. *Ind Eng Chem Res*. 1991;30:163–169.
35. Lee DS, Vassiliadis VS, Park JM. List-based threshhold-accepting algorithm for zero-wait scheduling of multiproduct batch plants. *Ind Eng Chem Res*. 2002;42:6579–6588.
36. Hui CW, Gupta A, Meulen H. A novel MILP formulation for short term scheduling of multistage multi-products batch plants with sequence-dependent constraints. *Comput Chem Eng*. 2000;24:1611–1617.
37. Lin B, Miler DC. Tabu search algorithm for chemical process optimization. *Comput Chem Eng*. 2004;28:2287–2306.
38. He Y, Hui CW. Self-learning approaches based on genetic algorithm for single-stage multi-product scheduling with compound objective. In: *Proceedings of the 17th International Congress of Chemical and Process Engineering (CHISA 2006)/9th Conference on Process Integration, Modeling and Optimization for Energy Saving and Pollution Reduction (PRES 2006)*, Praha, Czech Rpublic, 2006.

Manuscript received Nov. 28, 2006, and revision received May 19, 2007.