

Process Systems Engineering and Process Safety

Genetic algorithm for short-term scheduling of make-and-pack batch production process



Wuthichai Wongthatsanekorn*, Busaba Phruksaphanrat

Industrial Engineering Department, Faculty of Engineering, Thammasat University, Pathum-thani 12120, Thailand

ARTICLE INFO

Article history:

Received 19 August 2014

Received in revised form 6 March 2015

Accepted 25 April 2015

Available online 22 May 2015

Keywords:

Genetic algorithm

Ant colony optimization

Tabu search

Batch scheduling

Make-and-pack production

Forward assignment strategy

ABSTRACT

This paper considers a scheduling problem in industrial make-and-pack batch production process. This process equips with sequence-dependent changeover time, multipurpose storage units with limited capacity, storage time, batch splitting, partial equipment connectivity and transfer time. The objective is to make a production plan to satisfy all constraints while meeting demand requirement of packed products from various product families. This problem is NP-hard and the problem size is exponentially large for a realistic-sized problem. Therefore, we propose a genetic algorithm to handle this problem. Solutions to the problems are represented by chromosomes of product family sequences. These sequences are decoded to assign the resource for producing packed products according to forward assignment strategy and resource selection rules. These techniques greatly reduce unnecessary search space and improve search speed. In addition, design of experiment is carefully utilized to determine appropriate parameter settings. Ant colony optimization and Tabu search are also implemented for comparison. At the end of each heuristics, local search is applied for the packed product sequence to improve makespan. In an experimental analysis, all heuristics show the capability to solve large instances within reasonable computational time. In all problem instances, genetic algorithm averagely outperforms ant colony optimization and Tabu search with slightly longer computational time.

© 2015 The Chemical Industry and Engineering Society of China, and Chemical Industry Press. All rights reserved.

1. Introduction

A typical make-and-pack production problem occurs in processing plants that produces many products such as food and beverages, chemicals, and other products. A typical process starts from mixing process in a mixing tank. The mixture is transferred into multipurpose storage tanks of different capacities for a fixed period of time. Then this stored mixture is moved to packing lines. The final products in various formats are shipped to customers [1]. The objective is to determine an efficient production schedule that minimizes the makespan and satisfies customer demand. Currently, the planner of make-and-pack process finds it very difficult to determine the best short-term schedule. The planning becomes a challenging task because it involves many products and deals with many aspects such as multiple resources, partial equipment connectivity, sequence-dependent changeovers, storage time, transfer time and batch spitting. Honkomp *et al.* [1] explained a case study in details and provided data from the actual production process.

Researchers attempted to solve this problem using different techniques. For example, Fündeling and Trautmann [2] applied a priority-based heuristics by scheduling batches iteratively. The method was not effective because moderate problem could not be solved in

reasonable time. Baumann and Trautmann [3] developed an efficient continuous-time precedence-based mixed integer linear program embedding with sets of symmetry-breaking constraints. The results showed that adding symmetry-breaking constraints and preprocessing step could greatly reduce CPU time and solve small and moderate instances within reasonable CPU time.

In recent years, metaheuristic methods have been applied to solve many discrete manufacturing problems. The aim of this paper is to apply a genetic algorithm (GA) to solve large make-and-pack problem. The results will be compared by solution quality and computational time with ant colony optimization (ACO) and Tabu Search (TS).

2. Literature Review

Make-and-pack problem was studied from many points of view. Baumann and Trautmann [3] summarized the literature for this particular problem. Their review focused on the model with continuous representation of time and categorized network-based models into state-task-network (STN) and resource-task-network (RTN). They concluded that none of the models could accommodate all characteristics of make-and-pack problem. The most similar model was studied by Belaid *et al.* [4] for a shampoo-production process, but their approach could not be used for make-and-pack problem because of different characteristics. Most of the techniques for STN and RTN were based on mixed integer linear programming (MILP). For example, Günther *et al.* [5] developed

* Corresponding author.

E-mail address: wwuthich@engr.tu.ac.th (W. Wongthatsanekorn).

an MILP model for yogurt industry. Liu *et al.* [6] developed an MILP with hybrid approach between discrete and continuous representation of time. Kopanos *et al.* [7] solved scheduling problem with family-dependent changeover using MILP technique. Velez and Maravelias [8] solved chemical production scheduling problem using multiple and nonuniform time intervals with discrete-time mixed integer programming (MIP) models. They presented two different algorithms to find effective intervals and reduced the size of the formulation. As a result, larger problem could be solved with discrete-time MIP model. Velez and Maravelias [9] improved the efficiency of solving discrete-time model using parallel computing technique and solved each subproblem as an MIP on a single core of a computer. This technique could help solve more challenging problems than the default parallel option. Karimi-Nasab and Seyedhoseini [10] solved a job shop scheduling problem by using integer linear programming formulation. The performance of cut-and-branch and branch-and-cut approaches was shown on a set of randomly-generated test data.

The make-and-pack problem is similar to batch production scheduling studied by many researchers. For example, Yu *et al.* [11] provided an overview of optimization methods for batch scheduling and constructed a benchmark problem solved with different models. They suggested that reduction of a large number of binary variables would help solve large problems. Guohui *et al.* [12] modeled this problem as an MILP model to maximize profit. A line-up competition and linear programming were used to solve two examples in literature. Their results showed better improvement compared to the earlier work. Rabie and El-Halwagi [13] optimized the schedule of batch water recycle network. New source-tank-sink presentation and iterative approach were applied to solve case study problems. Metaheuristics such as GA was also applied by various researchers. For instance, Li *et al.* [14] combined GA with MILP to solve real batch problems. Wu *et al.* [15] developed GA and modified GA approaches to find a solution for case study problems. The results showed that modified GA featuring mixed coding of two different crossover and mutation operators were more effective than GA.

This make-and-pack problem can also be categorized as multiple-product multiple-resource scheduling problem (MMSP) with some exceptions. GA was applied to solve MMPS with some successes. He and Hui [16] implemented GA with forward or backward assignment strategy to solve large MMSP in batch plant. Their results showed significant reduction in search space and enhancement of search speed. He and Hui [17] later solved MMSP using binary encoding GA. With new proposed crossover, their technique could solve large instances within reasonable time. He and Hui [18] improved the original GA in [16] by proposing a novel global search framework. A comprehensive set of position selection rules was constructed and appropriate synthesis rules were applied. Capón-García *et al.* [19] dealt with multiobjective batch process scheduling problem *via* GA and local search and focused on the trade-off between environmental impact and economical throughput.

Some researchers focused on the case study proposed by [1]. Fündeling and Trautmann [2] used a priority-based heuristics to solve make-and-pack problem by scheduling batches iteratively. They proposed 287 multi-level priority rules for computing these values. Baumann and Trautmann [3] developed an efficient continuous-time precedence-based mixed-integer linear program with symmetry-breaking constraints and special preprocessing. They applied new technique to solve a set of medium scale problems. The results were up to 12% shorter than those from the heuristics of [2]. For a large scale problem, Fündeling and Trautmann [2] reported the minimum makespan of 7946 min on the first week data from [1].

Make-and-pack problem can be also considered as a classical job shop scheduling problem (JSP), where each product type has its own route on a set of resources and machines that are fixed and known in advance. It is well known that JSP problem is NP-hard [20]. It is very difficult to obtain optimality with typical optimization technique and metaheuristics to solve JSP problem [21,22]. Hence, make-and-pack problem in this paper is also NP-hard and no solution method can

ensure the global optimal for large scale problem with reasonable computational time. As it is shown in literature, GA has not been studied to solve large scale make-and-pack problem.

3. Problem Definition

For manufacturers of consumer goods industry, a typical make-and-pack process involves mixing tanks, storage tanks and packing lines. The final products are shipped to customers. High makespan means unnecessary labor cost [1]. The goal is to find the schedule that minimizes the makespan while satisfying all constraints. The details of this process are explained in Fig. 1.

- Step 1: For formula A, raw materials are sent to premix tank 1 (PM1). Premix process starts at “a” and finishes at “b” and the premix batch is transferred to main mix tank from “b” to “c”. For formula B, tank PM1 needs to be cleaned first, with a change-over time from “c” to “d”. Then, premix process starts at “d” and finishes at “j” and this premix batch is transferred to main mix tank from “j” to “k”.
- Step 2: For formula A, main-mix process starts production at “c” and finishes at “e” and main-mix batch is pumped out to single storage tank (ST1) from “e” to “f”. The capacity of ST1 is one batch, so one storage tank is required. For producing formula B, tank MM1 needs to be washed and there is a washout time of 47 min from “f” to “g”. Next, main-mix process starts production at “k” and finishes at “l” and main-mix batch is pumped out to two storage tanks (ST7 and ST8) from “l” to “m”. Since the capacity of both storage tanks is only 0.5 batches, two storage tanks are required. This batch splitting makes this problem more difficult to solve.
- Step 3: For bulk formula A, ST1 stores mixing batch from “f” to “g” and the final product is sent to PL1. For formula B, ST7 and ST8 store mixing batch from “m” to “n” and the final product is sent to PL1 (from ST7) and PL2 (from ST8).
- Step 4: For formula A, packing process starts at “g” and finishes at “h”. Task from ST8 is sent to PL1 to pack product from formula B. Packing change over time occurs at time “h–i”. PL1 then starts packing at time “n” and finish at time “o”. The process of wash out family is added from “o” to “p”.

The data for the case study are as follows.

3.1. Processing resources

There are three premix tanks and six main tanks with one-batch capacity each, six storage tanks with one-batch capacity and 74 tanks with 0.5-batch capacity, and seven packing lines supplying 0.5 batches. Furthermore, there is a restriction on the formula that premix and main mix tanks can process. Each product can be packed by a limited set of packing lines.

3.2. Demand of packed products

There are 203 packed products from 59 bulk formulas. Each formula can be packed into multiple container sizes. For product code “B5-V10-P13”, “B5-V10” represents formula name and “P13” is the packing code. Ten-week demand is given. The unit of demand is in batches and the size of batches is predetermined by processing equipment. It is assumed that due date is one week or 10080 min without a break. The requirement is to finish producing every packed product by the due date.

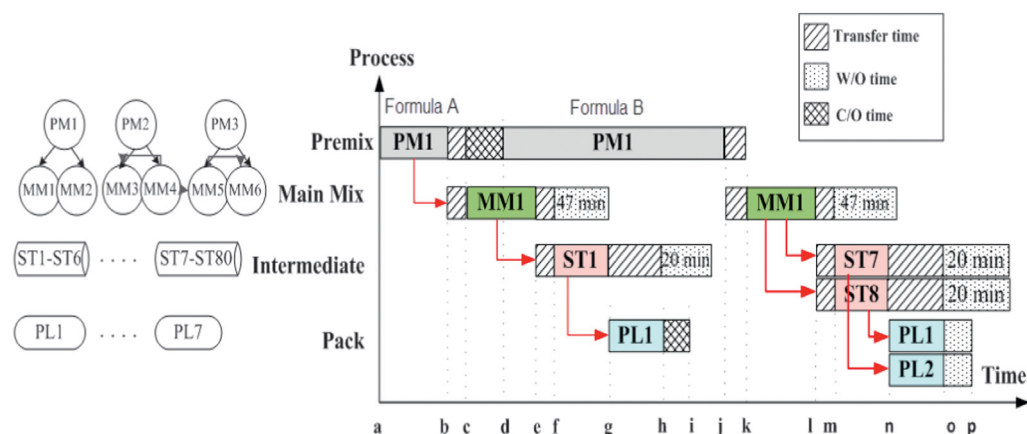


Fig. 1. Production, storage, and packing process of bulk formula A and B.

3.3. Changeover time

When a change of product or formula occurs, changeover time must be taken into account. This changeover is sequence-dependent because each product and formula has different characteristics. There are three types of changeover as follows.

- (1) Wash-out (W/O): Each formula is assigned to a wash-out family (U-1 to U-7). Wash-out occurs in all processes and it is applied when one wash-out family changes to another according to Table 1. If the value in Table 1 is 1, the wash-out happens when a product type changes. The wash-out time at premix and main mix tanks is always 47 min while it is always 20 min in storage tanks. For the packing lines, their wash-out time is different.
- (2) Packing changeover (C/O): This occurs at packing line when package type changes. Packing C/O time for each packing line is described in Table 1. For example, a switch from “B5-V10-P13” to “B5-V22-P9” at PL1 incurs only changeover time of 27 min because there is only one change in packing type from “P13” to “P9” while both “B5-V10” and “B5-V22” have the same wash-out family (U-7).
- (3) Combined wash-out and packing changeover (combined W/O & C/O): This occurs in packing process when it requires both wash-out and packing changeover. The time for each packing line is described in Table 1. For example, switching from “B5-V10-P13” to “B5-V27-P4” at PL2 incurs both wash-out and changeover time of 97 min, not 122 (87 + 35) min, because there is a change in packing type from “P13” to “P4”. Also, “B5-V10” and “B5-V27” have different wash-out family.

Table 1
Wash-out family relationship for all resources and wash out time (min)

	To U-1	To U-2	To U-3	To U-4	To U-5	To U-6	To U-7
From U-1	0	1	1	1	1	1	1
From U-2	0	0	1	1	1	1	1
From U-3	1	1	0	1	1	1	1
From U-4	0	0	1	0	1	1	1
From U-5	0	0	1	1	0	1	1
From U-6	1	1	1	1	1	0	1
From U-7	1	1	1	1	1	1	0

Type	PL1	PL2	PL3	PL4	PL5	PL6	PL7
W/O time	90	87	90	93	85	85	90
C/O time	27	35	33	40	30	35	40
Combined W/O & C/O time	105	97	106	110	95	95	108

3.4. Processing time

There are six types of processing time for the make-and-pack process. For “B1-V16”, the premix time, transfer time, main mix time, pump out time, and quality release time of storage tanks are 57, 11, 118, 18 and 48 min, respectively. According to Fig. 1, one batch of “B1-V16” spends 46 min at premix process and 11 min for the transfer to the main mix tank. Next, this batch spends 89 min in main mix process because it requires 11 min to transfer into the main mix tank and other 18 min to pump into storage tanks. At storage tanks, this batch always spends 45 min for quality release time after the batch is transferred into the storage tanks. When both premix time and transfer time are zero, the batch is processed in main mix tank only. The last type is the packing time, which depends on the combination of finished product and packing line.

Since the size of the realistic problem is very large, it is difficult to apply a typical mathematical programming approach. In this paper, we use GA, ant colony optimization and Tabu search to solve this NP-hard problem.

4. Solution Approach

4.1. Genetic algorithm (GA)

GA [23] is originated from Darwin's theory of evolution. This technique has been applied to solve various combinatorial optimization problems and it is found to be very effective. Solutions are represented by chromosomes. Each chromosome is evaluated through synthesizing into a possible schedule to compute fitness value. Many chromosomes are randomly generated at first. During crossover and mutation processes, new offspring is created from previous generation. The chromosome that survives from evolving many generations till the end of the iteration provides the near-optimal solution to the problem. The details of GA are explained next.

4.1.1. Solution representation

The first task is to represent a solution as a chromosome. A permutation based representation is applied here. Fig. 2 illustrates the solution representation through a small example with three product families

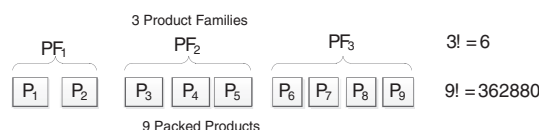


Fig. 2. Small example of solution representation.

and nine packed products. There are 2, 3 and 4 packed products for product family PF_1 , PF_2 , and PF_3 , respectively. When packed product-based sequence is used as the solution representation of chromosome, the total number of possible solutions is $9!$ (362880). However, when product family-based sequence is used instead, the number is reduced to $3!$ (6).

Thus we use the product family-based sequence to represent the chromosome. We also assume that the packed product with the same product family should be produced together. According to [1], there are 59 product families and 203 packed products, so it is easier to solve a problem with $59!$ possible solutions (1.39×10^{80}) rather than $203!$ possible solutions (6.05×10^{381}). We also need to know the sequence of packed products for each product family to evaluate the fitness function. We sort each packed product by packing type first and by demand quantity in descending order later. For M product families ($PF_1, PF_2, \dots, PF_M$) with each product family containing np_i ($i = 1, 2, \dots, M$) packed products ($P_1, P_2, \dots, P_{np_i}$) and N total packed products, a chromosome C for GA is the sequence of M product families instead of N packed products. The size of a chromosome is equal to M as shown in Fig. 3.

This chromosome representation is different from that in [16] for MMSP problem. For make-and-pack problem, the sequence of product family production is the same for every resource stage while the sequence may be different in [16]. Therefore, when the chromosome is synthesized into schedule, it is not evaluated stage by stage but each order is assigned to the resource starting from premix stage to packing stage before the next order is evaluated. This is because the resources in each stage are connected to each other with transfer time between them. Next, the fitness function evaluation of each chromosome is explained.

4.1.2. Evaluation of chromosome

The chromosome, C , only represents the sequence of production, not an actual schedule. This subsection explains a procedure to synthesize the chromosome into a schedule in order to evaluate the fitness value, $F(C)$. For make-and-pack problem, the objective is to minimize the makespan or finishing time of the last pack-product in the packing process. This synthesized technique follows the work by [16].

4.1.2.1. Forward assignment strategy. To compute the makespan, the schedule is generated by assigning resource in each stage according to the sequence of production with forward assignment strategy technique [16]. The resource unit with the earliest available processing time is assigned first. If there is a tie, the resource is chosen upon the resource selection rules. The idea behind this assignment is that the packing process of any products should be finished before making new resource assignment for new product family. Fig. 4 illustrates the application of this strategy for chromosome, $C = [PF_1, PF_2]$. The schedule is obtained by assigning all necessary resources (PM_1, MM_1, ST_1, PL_1) for PF_1 to pack product P_1 . Once this is finished, the available time of all resources is adjusted accordingly. Next, we assign resources ($PM_1, MM_1, ST_1, ST_2, PL_1, PL_2$) for PF_2 to pack product P_2 . Consequently, the makespan could be calculated.

He and Hui [16] have also synthesized the chromosome into active schedule by assuming that the job order at each resource stage could be different. Kreutz *et al.* [24] have stated that active schedule contains no excel idle time and no operation can be completed earlier without delaying other operations. The schedule for make-and-pack problem is also active because (1) each gene or order will be processed by

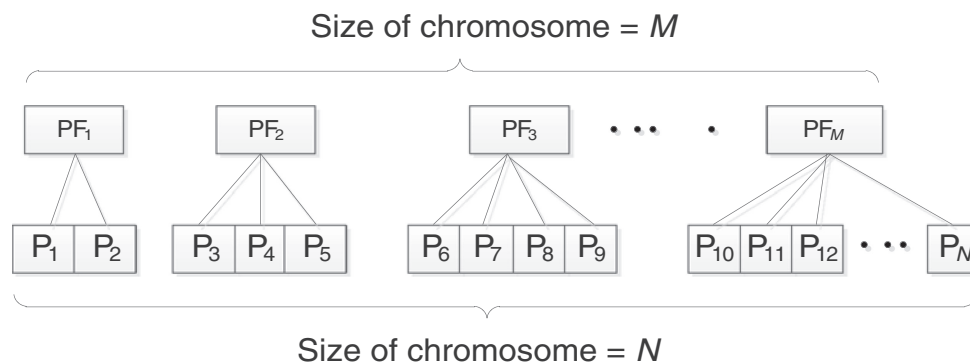


Fig. 3. Solution representation of all product families vs. packed products.

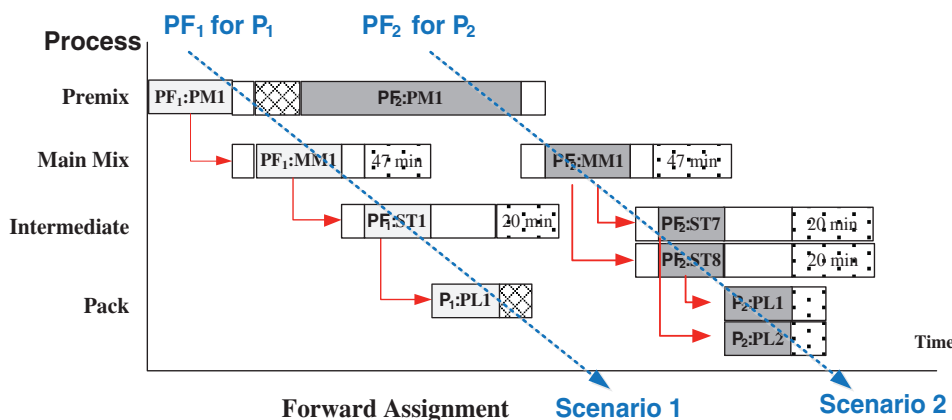


Fig. 4. Forward assignment strategy concept.

assigning resource unit in premix, main mix, storage, and packing line, (2) each resource is connected, and (3) there is transfer time between resource units. Therefore, this schedule will not have any excess idle time.

4.1.2.2. Resource selection rules. In make-and-pack problem, there may be several parallel resource units available for assignment. Resource selection rules are required to handle this situation. For premix and main mix tanks, the earliest assignment rule can be applied directly because both tanks have the same capacity of one batch and the connection between these two stages is one-to-one. The complication occurs at the storage tanks because some of them have capacity of one batch while others have the capacity of 0.5 batches. Hence, two scenarios could happen.

Scenario 1: product from a main mix tank is transferred to a one batch storage tank.

Scenario 2: product from a main mix tank is transferred to two half batch storage tanks.

For the first scenario, the earliest available one-batch storage tank is chosen to produce PF_1 . In the second scenario, the batch is split into half to produce PF_2 as shown in Fig. 4. The resource selection rules are as follows.

- (1) If the first available storage tank has one-batch capacity, that storage tank is chosen.
- (2) If there are two half batch storage tanks with the same earliest available time, the batch is split in half and transferred to both storage tanks.
- (3) If only one half batch storage tank is available, the rest of the batch is assigned to the next available half batch tank, not one batch tank.
- (4) If one batch tank and at least two half batch tanks have the same earliest available time, each scenario is randomly assigned with the same probability (0.5).

From the preliminary results, the choice of scenarios 1 and 2 affects the makespan of the schedule. Sometimes it is better to wait for a one-batch tank instead of spitting the batch into half. Hence, we make another selection rule: (5) if the difference between the earliest available time of half batch tanks and next available time of one batch tank is less than 20 min, the probability that one batch tank is assigned equals to 0.5. This increases the chance of finding better schedule.

4.1.3. Procedure of genetic algorithm

The procedure of GA can be described in the following steps.

- Step 1. Initialization: randomly generate solutions or chromosomes equal to the population size (popsize). These chromosomes are evaluated for makespan through synthesizing the chromosomes into schedules. Two sets of parameters are considered. (1) popsize = 100 and number of iterations = 50; (2) popsize = 50 and number of iterations = 100. In each iteration, a new generation is created through selection, crossover and mutation operations while the population size stays the same.
- Step 2. Selection: select a pair of parents based on the selection technique. We use roulette-wheel selection (RWS) and ranked-based roulette wheel selection (RRWS). For RWS [25], the probability for choosing any chromosome is equal to $(\bar{F}_i / \sum \bar{F}_i, \bar{F}_i = \max_i (F_i) - F_i)$. The chromosome with the highest fitness value has a higher chance of being selected because of better fitness value. For RRWS, Jadann *et al.* [26] and Razali and Geragty [27] have showed that this selection

method can offer a better result than RWS for traveling salesman problem because selection probabilities are computed based on the rank instead of the fitness values. Rank is scaled learning from the following formula, $Rank = 2 - SP + (2(SP - 1)(pos - 1)/(M - 1))$, where SP is selective pressure (set to 1.1) and pos is the position of chromosome based on the fitness value. The chromosome with the least makespan has a value of pos set to M .

- Step 3. Crossover: generate offspring with crossover technique using crossover probability. If none takes place, use exact copies of the parents in the next step. Union crossover #2 (UX2) and order-based crossover (OX2) methods are selected. Poon and Carter [28] have compared the performance of several crossover operators and suggested UX2 for solving traveling salesman problem and other ordering problems. For UX2, we let the first parent be 5 1 3 2 6 4 and the second parent be 2 4 1 6 3 5. Then, three genes (4 1 6) are randomly chosen from the first parent (S1). Other three genes (5 3 2) are randomly chosen from the second parent (S2). The offspring is generated from randomly chosen S1 or S2 in each ordering position. If the ordering position of the offspring is S2 S2 S1 S1 S1 S2, the obtained offspring is 5 3 4 1 6 2.

Larranaga *et al.* [29] have tested many crossover methods for ordering problems and concluded that OX2 is a second-best operator but runs faster than the best operator. OX2 selects random positions of one parent and it is imposed on the other parent. If the first and second parents are (1 2 3 4 5 6) and (2 4 6 5 3 1), the second, third and sixth positions of the second parents are chosen and represented as 4, 6, and 1, respectively. These selected elements represent the first, fourth and sixth positions of the first parent. The offspring is set equal to first parent except the first, fourth and sixth positions: (* 2 3 * 5 *). The missing elements are added to the offspring in the same order, which appear in the second parent. The offspring is then (4 2 3 6 5 1). If the roles of the first and second parents exchange, the other offspring is (2 4 3 5 6 1).

- Step 4. Mutation: perform mutation operation on the new offspring using mutation probability. There are various mutation techniques [29] such as displacement, exchange, insertion, simple-inversion, inversion and scramble. It is shown [29] that simple inversion (SIM) and exchange mutation operators (EM) provide good results for ordering problem similar to our make-and-pack problem. Furthermore, these two operators are easy to implement and require little computational time since only two random positions are generated. For SIM, two cut points are randomly selected and the elements between them are reversed. For example, if the chromosome is (1 2 3 4 5 6) and the two cut points are at second and fifth positions, the resulting offspring is (1 5 4 3 2 6). For EM, two elements are randomly chosen and their values are exchanged. For example, if the chromosome is (1 2 3 4 5 6) and the two elements are at second and fifth positions, the resulting offspring is (1 5 3 4 2 6).

- Step 5. Check the termination condition: compute makespan of new offspring and sort all chromosomes. Choose best popsize chromosome for the next iteration. If the number of iteration reaches the limit, stop; otherwise, go back to step 2. If the best solution remains the same for 50 consecutive iterations, GA also terminates.

The design of experiment is used to investigate appropriate setting of GA parameters (a combination of population size and the number of generation (P/G), selection operator (SOP), crossover operator (COP), mutation operator (MOP), probabilities of crossover ($P_C\%$) and mutation ($P_M\%$)). All parameters are investigated at two levels. This

allows the relationship between the response and the design factors to be modeled as 2-factor full-factorial design. The details are given in Table 2.

Table 2
Parameter setting of all factors

Factor	Level
Population size/number of iteration (P/G)	50/100, 100/50
Selection operator (SOP)	RRWS/RWS
Crossover operator (COP)	UX2, OX2
Mutation operator (MOP)	SIM/EM
Crossover probability (P_C /%)	0.8, 0.9
Mutation probability (P_M /%)	0.1, 0.2

If the running time is not limited, the chance of finding an optimal solution increases with higher population size and number of iteration. These two parameters decide the number of generated chromosomes for GA. With higher values of these two parameters, larger search space is covered. The combination of population size and number of iteration is fixed at 5000 generated chromosomes because the solutions converge in the preliminary study. From the number of factors and their levels, a full factorial (2^6) is adopted with five replications. The data for this experiment is the large-scale problem [1] using the demand of week 1.

The results are analyzed using the general linear form of analysis of variance (ANOVA) shown in Table 3. The factors with p-value less than 0.05 are statistically significant with 95% level of confidence. For the main effect, P/G, SOP and COP are statistically significant with 95% confident level. MOP is almost statistically significant at p-value 0.061. For the 2-way interaction, (P/G) $\times$ COP, (P/G) $\times$ MOP, SOP $\times$ COP, and SOP $\times$ MOP are statistically significant with 95% confident level. To find suitable parameter setting, response surface optimizer is applied and appropriate settings are shown in Fig. 5. The appropriate levels of factor P/G, SOP, and COP are 50/100, RWS, OX2, respectively. Although MOP, P_C , and P_M factors are not theoretically significant, appropriate levels must be specified in order to run GA to solve make-and-pack problem. The results from the response surface optimizer show that MOP, P_C and P_M should be set to EM, 0.9 and 0.2, respectively.

Table 3
ANOVA Table from design of experiment

Source	DF	Sum of squares	Mean square	F	p-Value
Main effects	6	655000	109167	48.87	0.000
P/G	1	11305	11305	5.06	0.025
SOP	1	272403	272403	121.95	0.000
COP	1	361299	361299	161.75	0.000
MOP	1	7917	7917	3.54	0.061
P_C	1	1748	328	0.78	0.377
P_M	1	328	5795	0.15	0.702
2-Way	15	86929	1933	2.59	0.001
(P/G) $\times$ SOP	1	1933	10870	0.87	0.353
(P/G) $\times$ COP	1	10870	11035	4.87	0.028
(P/G) $\times$ MOP	1	11035	3136	4.94	0.027
(P/G) $\times$ P_C	1	3136	835	1.4	0.237
(P/G) $\times$ P_M	1	835	29263	0.37	0.541
SOP $\times$ COP	1	29263	16188	13.1	0.000
SOP $\times$ MOP	1	16188	1061	7.25	0.008
SOP $\times$ P_C	1	1061	3956	0.47	0.494
SOP $\times$ P_M	1	3956	2638	1.77	0.184
COP $\times$ MOP	1	2638	4	1.18	0.278
COP $\times$ P_C	1	4	441	0	0.968
COP $\times$ P_M	1	441	965	0.2	0.657
MOP $\times$ P_C	1	965	4506	0.43	0.512
MOP $\times$ P_M	1	4506	98	2.02	0.157
$P_C \times P_M$	1	98	2234	0.04	0.834
Residual error	298	665656			
Total	319	1407585			

4.2. Ant colony optimization (ACO)

ACO heuristics is introduced by [30,31]. This heuristics imitates how ants find their food source. Each solution or product family sequence is represented by each ant. Initial step begins with constructing M ants according to state transition rule. In each iteration, amount of pheromone is updated with global updating rule. Ants or solutions move to the next solution via both pheromone updated amount and heuristic information to obtain the best solutions. The heuristics terminates after 100 iterations.

The state transition rule is to move the solution with probability $p_{ij}^k(t)$, with ant or solution k producing product family i in j th production order in iteration t ,

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{j=1}^{PF} [\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta} \quad (1)$$

where τ_{ij} is the pheromone intensity, α is the relative importance of trail, and β is the relative importance of heuristic information. If α is set to 0, product family pair (i, j) with minimum change/over time will be selected. If β is set to 0, all the ants will follow the same path and construct the same solution. Hence, there is a tradeoff between α and β . Heuristic information for producing product family j after product family i (η_{ij}) is

$$\eta_{ij} = \frac{1}{CO_{ij}} \quad (2)$$

where CO_{ij} is change/over time for switching to product family j after product family i . For the pheromone updating rule, the trail intensity can be updated as

$$\tau_{ij}(\text{new}) = \rho \tau_{ij}(\text{old}) + \sum_{k=1}^M \Delta \tau_{ij}^k \quad (3)$$

where ρ is a coefficient such that $1 - \rho$ is evaporation of trail, which is used to avoid unlimited accumulation of pheromone trails. For $\Delta \tau_{ij}^k$, it is defined as

$$\Delta \tau_{ij}^k = \begin{cases} \text{Makespan}^k(t) & \text{if the } k\text{th ant produce product family } j \text{ after } i \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $\text{Makespan}^k(t)$ is the makespan of the k th ant's production sequence. This means that the better the makespan, the more pheromone is received by an arc that represents producing product family j after i . After many computation runs, the best parameter for solving make-and-pack problem is achieved by setting $m = 50$, $\alpha = 0.5$, $\beta = 0.8$, and $\rho = 0.02$.

4.3. Tabu search (TS)

TS heuristics, implemented by [32,33], is basically a local search. It is performed around the neighborhood of current solution in all iterations. The best solution found in the neighborhood is set as new current solution. The procedure is terminated after 100 iterations. For TS, a Tabu list is created to prohibit some neighborhood. In the first step, a good feasible solution is found using greedy approach. Fifty solutions are randomly generated and the best solution is used as an initial solution. A neighborhood of the initial solution is found using SIM operator. Two product families are randomly selected and exchanged. Tabu list contains a pair of product families that are previously exchanged. The rule is that it is not possible to swap the pair in the Tabu list. The Tabu structure stores the number of time with a given pair prohibited from the exchange. If a pair is prohibited for fixed restriction period, that pair is removed from the Tabu List. This is called aspiration criterion. This process is repeated until the number of iterations exceeds the limit. If the

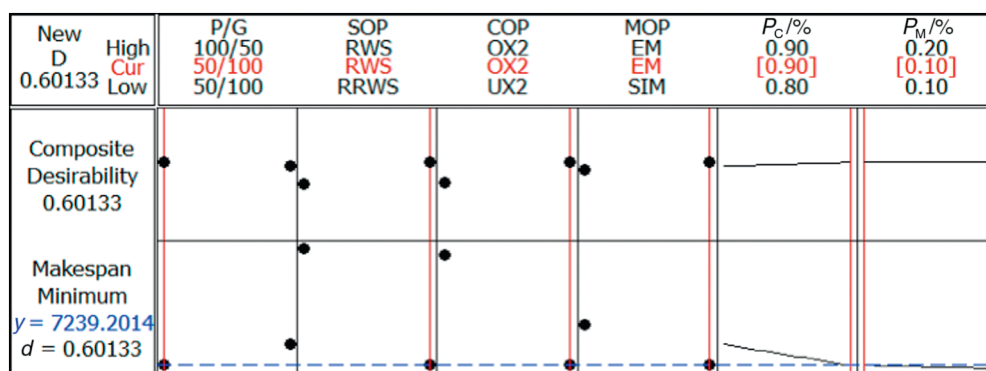


Fig. 5. Optimization setting using response surface optimizer.

solution has not improved by given frequency limit, new randomly generated solution is used as the current solution. After many computation runs, the best parameter for solving make-and-pack problem can be achieved by setting the length of Tabu list to 59, restriction period to 9, and frequency limit to 15. To make the same comparison with GA and ACO, TS is repeated 50 times and the best solution is reported.

4.4. Local search

In order to improve the quality of solution, local search technique is applied at the end of GA, ACO and TS procedures. Each heuristics provides the best production sequence of product family. The sequence of packed products of each product family is sorted by demand quantity in descending order. There is no guarantee that this sequence of packed product returns the minimum makespan. Hence, it is possible to find a better solution in the neighborhood of the last obtained solution.

The idea is to fix the sequence of product family but vary the sequence of packed products in each product family. Three types of local search are implemented. They are simple inversion, exchange mutation operators and random search with the sequence of packed products shuffled randomly. The goal is to diversify the search space and look for better solution. Each local search is applied for 50 times. At the end of the local search, the best solution is reported.

5. Experimental Results

GA, ACO and TS are applied to solve ten large problems from the case study of Honkomp *et al.* [1] using the demand from week 1 to 10. The heuristics is implemented in C# language and the results are run on a PC with 2300 MHz CPU and 4 GB memory. Each problem is solved ten times. Minimum, maximum and mean makespan are computed and reported in Table 4, with standard deviation (SD), coefficient of variation (CV = $100 \times \text{SD}/\text{mean}$), absolute mean deviation from best, and

Table 4
Results of GA, ACO and TS for large problems

Week	Method	Min	Max/min	Mean/min	SD/min	CV/%	Absolute mean deviation from best/min	Percentage of mean deviation from best/%	Average CPU time/s
1	GA	7290.63	7464.27	7382.10	47.99	0.65	91.47	1.25	7.85
	ACO	7290.63	7487.65	7402.54	49.02	0.66	95.54	1.31	4.87
	TS	7503.82	7738.62	7638.21	49.57	0.65	94.24	1.26	5.21
2	GA	7560.12	7856.03	7679.55	94.67	1.23	119.43	1.58	8.34
	ACO	7653.56	7948.08	7808.47	99.04	1.27	130.21	1.70	5.87
	TS	7610.65	7951.47	7816.59	120.00	1.54	147.24	1.93	4.58
3	GA	7498.50	7666.15	7571.94	55.02	0.73	73.44	0.98	7.98
	ACO	7532.22	7688.55	7626.21	59.00	0.77	89.54	1.19	4.87
	TS	7759.76	7785.54	7784.13	56.00	0.72	72.54	0.93	6.12
4	GA	7562.49	7847.06	7730.48	85.66	1.11	167.99	2.22	7.89
	ACO	7663.36	7955.40	7821.22	86.24	1.10	164.25	2.14	5.67
	TS	7689.65	7956.54	7839.38	125.49	1.60	195.21	2.54	5.24
5	GA	7541.89	7990.53	7684.72	142.25	1.85	142.83	1.89	7.86
	ACO	7609.03	8003.51	7706.53	144.25	1.87	148.02	1.95	5.14
	TS	7793.49	8157.45	7911.57	156.70	1.98	198.24	2.54	5.87
6	GA	7259.17	7465.59	7389.40	60.86	0.82	130.23	1.79	7.96
	ACO	7342.54	7553.70	7480.33	63.77	0.85	135.24	1.84	4.54
	TS	7610.65	7610.65	7633.84	88.01	1.15	187.24	2.46	5.04
7	GA	7830.83	8088.12	7936.08	79.94	1.01	105.25	1.34	8.35
	ACO	7830.83	8104.23	7957.42	80.83	1.02	107.24	1.37	4.16
	TS	8008.89	8293.93	8191.21	82.78	1.01	105.27	1.31	4.89
8	GA	7393.30	7556.81	7488.88	50.51	0.67	95.58	1.29	7.66
	ACO	7424.78	7565.56	7518.04	55.47	0.74	100.24	1.35	4.25
	TS	7610.65	7680.65	7626.51	54.96	0.72	96.24	1.26	5.57
9	GA	6534.00	6672.75	6610.57	42.11	0.64	76.57	1.17	7.54
	ACO	6531.73	6696.97	6674.54	44.87	0.67	78.24	1.19	6.01
	TS	6698.59	6839.09	6800.81	44.80	0.66	77.59	1.16	5.34
10	GA	7475.10	7884.22	7650.34	109.46	1.43	175.24	2.34	8.41
	ACO	7480.25	7915.56	7675.50	110.16	1.44	176.25	2.36	4.57
	TS	7610.65	7954.54	7714.54	163.54	2.12	201.5	2.65	5.78
AVG	GA	7394.60	7649.15	7512.41	76.85	1.01	117.80	1.59	7.98
	ACO	7435.89	7691.92	7567.08	79.26	1.04	122.48	1.64	5.00
	TS	7589.68	7796.85	7695.68	94.19	1.21	137.53	1.81	5.36

Table 5

Local search result comparison of GA, ACO and TS for large problems

Method	Mean/min	Average CPU time/s	Method	Mean/min	Average PU time/s	Method	Mean/min	Average PU time/s
GA	7614.58	7.67	ACO	7651.83	4.80	TS	7771.10	5.39
GA with LS	7512.41	7.98	ACO with LS	7567.08	5.00	TS with LS	7695.68	5.59
Difference	102.17	0.32	Difference	84.751	0.20	Difference	75.42	0.23
Difference percentage	1.34%	4.08%	Difference percentage	1.11%	4.16%	Difference percentage	0.97%	4.29%

percentage mean deviation from best. The results explain the robustness of three heuristics on make-and-pack for large-scale problems. In these problem instances, the values are just between 0.64%–2.12% and the percentage mean deviation from best is 0.93%–2.65%. The values of CV and percentage mean deviation from best in all instances are less than 3%, so the robustness in obtaining the optimal solution for large instances is effective for all these methods. Furthermore, the average CPU time is only 4.16 to 8.41 s. Hence, all three methods are effective in solving large instance of make-and-pack problems.

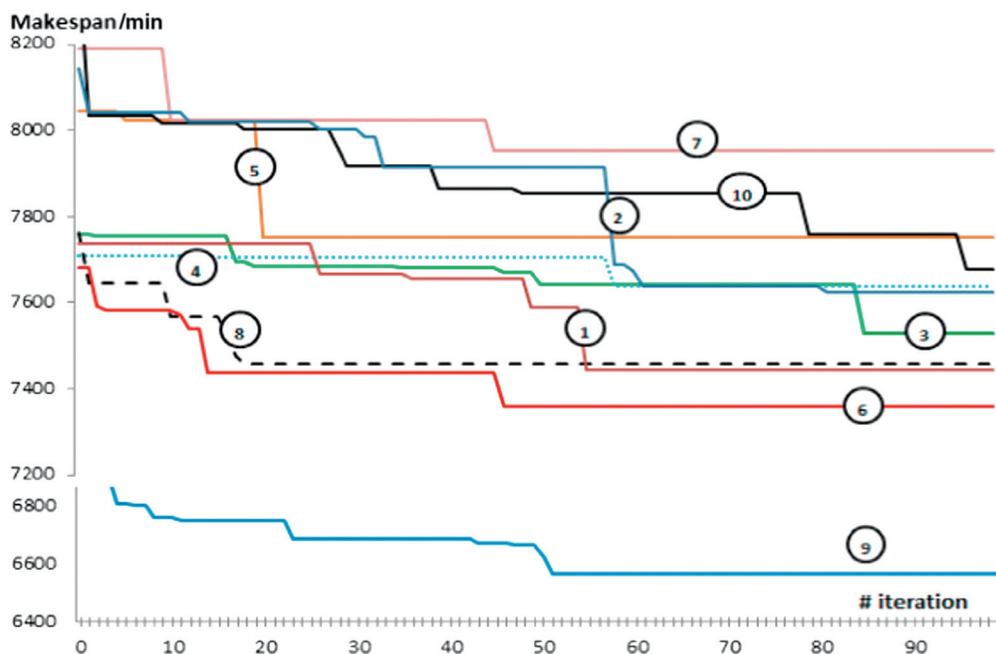
The minimum makespan of each heuristics is analyzed. The minimum makespan in the first week is 7290.63 min obtained by GA and ACO. This is 8.13% lower than what Fündeling and Trautmann [2] found at 7936 min. In almost all instances except week 9, solutions from GA are less than or equal to solutions from ACO and TS. According to Table 4, the minimum makespan in all ten weeks from GA is 7394.60 min, which is lower than that from ACO and TS, at 7435.89 min and 7589.68 min, respectively. Thus, the minimum makespan by GA is better than that by ACO and TS, on average by 0.56% and 2.07% respectively. The results are similar for average maximum makespan and mean makespan. Compared to that of ACO and TS, the average maximum makespan with GA is smaller by 0.56% and 1.36%, and its average mean makespan is better by 0.73% and 1.70%, respectively. On the opposite, solutions from TS are either greater than or equal to those from GA and ACO. Hence, GA is averagely better than ACO and TS while TS is worse than GA and ACO.

A possible explanation is that GA combines both exploration and exploitation through crossover and mutation operations and works well in this make-and-pack problem where the fitness function is complex and the search space is extremely large. For ACO, the exploration only occurs

at the beginning when M ants are randomly generated. The rest of the procedure focuses on the exploitation to search for local optimal solution. For TS, only one type of neighborhood is applied and it is easier to get trapped into the local optimum through exploitation even with frequency limit option. From Table 5, the average CPU time of GA is the highest and that of TS is the lowest because of the nature of heuristics. This suggests that GA spends more time for searching a solution. In summary, since the solution quality of GA is averagely better, it is worth the computational time to search better makespan.

In all heuristics, local search helps find better solution. In Table 5 the average makespan from 10 weeks with local search shows the reduction by 1.34%, 1.11%, and 0.97% for GA, ACO and TS, respectively. However, the average computational time is increased by 4.04%, 4.16%, and 4.29% for GA, ACO and TS respectively. From the results, the only disadvantage of local search is slight increase of computational time, which is less than two seconds. Therefore, local search is beneficial in searching better solutions. In the real situation, quality of solution is more important than computational time because reduction of makespan means cost saving for the company.

Since GA outperforms other two heuristics, the effectiveness of GA to find optimal solution is further analyzed and shown in Fig. 6, which displays the best-solution in 100 iterations for large problem week 1 to 10 from the best replication. In problems 1, 4, 5, 6, 7, 8 and 9, the optimal solution converges in less than 60 iterations while the optimal solution for problems 2, 3 and 10 is obtained after 80 iterations. This is because the problem size is very large and it requires more iteration to find the best solution. The percentage improvement from initial solution is from 0.89% to 7.51%. Note that Fig. 6 does not show the improved answer by using local search.

**Fig. 6.** Convergence of GA for large instances.

6. Conclusions

In this paper, we present GA, ACO and TS approaches to solve short-term make-and-pack scheduling problem. This problem contains many complex features such as sequence-dependent changeover time, storage time, batch splitting, partial equipment connectivity and multiple storage units with different capacities. Solutions to the problems are represented by chromosomes of product family sequences. With forward assignment strategy and resource selection rules to convert the solutions to real schedules, GA, ACO and TS are applied to solve large production process problem. The results for large-sized problems from the case study of Honkomp *et al.* [1] are reported. From the computation results, all three methods are effective and robust to solve make-and-pack problem in all sizes. Moreover, by applying local search, the solution could be improved further with a small increase in the computational time. In all problem instances, genetic algorithm could find either the same or better solutions than ant colony optimization and Tabu search with slightly longer running time.

Future research includes adjusting the synthesized technique to cover more possibilities. Applying other improvement techniques to solve make-and-pack problem could be explored. Adding total changeover time and number of changeovers to the current objective, makespan, could also add another aspect of make-and-pack problem.

Acknowledgments

We acknowledge a sponsorship of Thailand Research Fund (Grant #MRC5480176) and National Research University Project of Thailand Office of Higher Education Commission.

References

- [1] S. Honkomp, S. Lombardo, O. Rosen, J.F. Pekny, The curse of reality—why process scheduling optimization problems are difficult in practice, *Comput. Chem. Eng.* 21 (1) (2000) 123–136.
- [2] C.U. Fündeling, N. Trautmann, Scheduling of make and pack plants: a case study, 16th International European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering, Elsevier, Garmisch-Partenkirchen 2006, pp. 1551–1556.
- [3] P. Baumann, N. Trautmann, A continuous-time MILP model for short-term scheduling of make-and-pack production processes, *Int. J. Prod. Res.* 51 (6) (2013) 1707–1727.
- [4] R. Belaid, V. T'Kindt, C. Esswein, Decomposition algorithms for planning the production of a real shampoo industry, International Conference on Industrial and Systems Management, Metz, France, 2011.
- [5] H.O. Günther, M. Grunow, U. Neuhaus, Realizing block planning concepts in make-and-pack production using MILP modeling and SAP APO, *Int. J. Prod. Res.* 44 (18–19) (2006) 3711–3726.
- [6] S. Liu, J. Pinto, L. Papageorgiou, A TSP-based MILP model for medium-term planning of single-stage continuous multiproduct plants, *Ind. Eng. Chem. Res.* 47 (20) (2008) 7733–7743.
- [7] G.M. Kopanos, L. Puigjaner, C. Maravelias, Production planning and scheduling of parallel continuous processes with product families, *Ind. Eng. Chem. Res.* 50 (3) (2011) 1369–1378.
- [8] S. Velez, C.T. Maravelias, Multiple and nonuniform time grids in discrete-time MIP models for chemical production scheduling, *Comput. Chem. Eng.* 53 (1) (2013) 70–85.
- [9] S. Velez, C.T. Maravelias, A branch-and-bound algorithm for the solution of chemical production scheduling MIP models using parallel computing, *Comput. Chem. Eng.* 55 (8) (2013) 28–39.
- [10] M. Karimi-Nasab, S.M. Seyedhoseini, Multi-level lot sizing and job shop scheduling with compressible process times: a cutting plane approach, *Eur. J. Oper. Res.* 231 (3) (2013) 598–616.
- [11] Y. Qian, M. Pan, Y. Huang, Modeling and optimization for scheduling of chemical batch, *Chin. J. Chem. Eng.* 17 (1) (2009) 1–7.
- [12] G. Chen, L. Yan, B. Shi, Modeling and optimization for short-term scheduling of multipurpose batch plants, *Chin. J. Chem. Eng.* 22 (6) (2014) 682–689.
- [13] A.H. Rabie, M.M. El-Halwagi, Synthesis and scheduling of optimal batch water-recycle networks* supported by the Texas Water Resources Institute (TWRI) and the Texas Hazardous Waste Research Center, *Chin. J. Chem. Eng.* 16 (3) (2008) 474–479.
- [14] L. Chen, K. Wang, X. Xu, P. Yao, Hierarchical on-line scheduling of multiproduct batch plants with a combined approach of mathematical programming and genetic algorithm, *Chin. J. Chem. Eng.* 12 (1) (2004) 78–84.
- [15] L. Wu, Y. Hu, D. Xu, B. Hua, Genetic algorithm-based approach to scheduling of batch production with maximum profit, *Chin. J. Chem. Eng.* 13 (1) (2005) 68–73.
- [16] Y. He, C.-W. Hui, Genetic algorithm for large-size multi-stage batch plant scheduling, *Chem. Eng. Sci.* 62 (2007) 1504–1523.
- [17] Y. He, C.-W. Hui, A binary coding genetic algorithm for multi-purpose process scheduling: a case study, *Chem. Eng. Sci.* 65 (16) (2010) 4816–4828.
- [18] Y. He, C.-W. Hui, A novel search framework for multi-stage process scheduling with tight due dates, *AIChE J.* 56 (8) (2010) 2103–2121.
- [19] E. Capón-García, A.D. Bojarski, A. Espuña, L. Puigjaner, Multiobjective evolutionary optimization of batch process scheduling under environmental and economic concerns, *AIChE J.* 59 (2) (2013) 429–444.
- [20] M.R. Garey, D.S. Johnson, R. Sethi, The complexity of flowshop and job-shop scheduling, *Math. Oper. Res.* 1 (1976) 117–129.
- [21] P. Fattahi, M. Saidi Mehrabad, F. Jolai, Mathematical modeling and heuristic approaches to flexible job shop scheduling problems, *J. Intell. Manuf.* 18 (3) (2007) 331–342.
- [22] W. Cheung, H. Zhou, Using genetic algorithms and heuristics for job shop scheduling with sequence-dependent setup times, *Ann. Oper. Res.* 107 (1–4) (2001) 65–81.
- [23] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.
- [24] M. Kreutz, D. Handke, S. Gehlen, Solving extended hybrid-flow-shop problems using active schedule generation and genetic algorithms, Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science, vol. 1917, Springer, Berlin 2000, p. 293–202.
- [25] D.E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
- [26] O.A. Jadann, L. Rajamani, C.R. Rao, Improved Selection Operator for GA, *J. Theor. Appl. Inf. Technol.* 4 (4) (2008) 269–277.
- [27] N.M. Razali, J. Geraghty, Genetic algorithm performance with different selection strategies in solving TSP, Proceedings of the World Congress on Engineering 2011 Vol II, WCE 2011, London, UK, 2011.
- [28] P.W. Poon, J.N. Carter, Genetic algorithm crossover operators for ordering applications, *Comput. Oper. Res.* 22 (1) (1995) 135–147.
- [29] P. Larranga, C.M.H. Kuijpers, R.H. Murga, Y. Yurramendi, Learning Bayesian network structures by searching for the best ordering with genetic algorithm, *IEEE Trans. Syst. Man Cybern. Syst. Hum.* 26 (4) (1996) 487–493.
- [30] M. Nourelfath, N. Nahas, Quantized hopfield networks for reliability optimization, *Reliab. Eng. Syst. Saf.* 81 (2003) 191–196.
- [31] N. Nahas, M. Nourelfath, Ant system for reliability optimization of a series system with multiple-choice and budget constraints, *Reliab. Eng. Syst. Saf.* 87 (2005) 1–12.
- [32] F. Glover, Tabu Search Part I, *ORSA J. Comput.* 1 (3) (1989) 190–206.
- [33] F. Glover, Tabu Search Part II, *ORSA J. Comput.* 2 (1) (1990) 4–32.