Int J Adv Manuf Technol (2008) 36:959–968
DOI 10.1007/s00170-006-0904-9

ORIGINAL ARTICLE

Hybrid evolutionary algorithm with marriage of genetic algorithm and extremal optimization for production scheduling

Yu-Wang Chen · Yong-Zai Lu · Gen-Ke Yang

Received: 5 September 2006 / Accepted: 29 November 2006 / Published online: 4 January 2007

Abstract This paper presents a hybrid evolutionary algorithm with marriage of genetic algorithm (GA) and extremal optimization (EO) for solving a class of production scheduling problems in manufacturing. The scheduling problem, which is derived from hot rolling production in steel industry, is characterized by two major requirements: (i) selecting a subset of orders from manufacturing orders to be processed; (ii) determining the optimal production sequence under multiple constraints, such as sequence-dependant transition costs, non-execution penalties, earliness/tardiness (E/T) penalties, etc. A combinatorial optimization model is proposed to formulate it mathematically. For its NP-hard complexity, an effective hybrid evolutionary algorithm is developed to solve the scheduling problem through combining the population-based search capacity of GA and the fine-grained local search efficacy of EO. The experimental results with production scale data demonstrate that the proposed hybrid evolutionary algorithm can provide superior performances in scheduling quality and computation efficiency.

Keywords Hybrid evolutionary algorithm · Genetic algorithm · Extremal optimization · Production Scheduling

Abbreviations

GA	genetic algorithm
MGA	modified genetic algorithm
EO	extremal optimization
GEO	genetic extremal optimization
E/T	earliness/tardiness
PCTSP	prize collecting traveling salesman problem

Y.-W. Chen (✉) · Y.-Z. Lu · G.-K. Yang
Department of Automation, Shanghai Jiaotong University,
Shanghai 200240, People's Republic of China
e-mail: cywpeak@sjtu.edu.cn

Y.-Z. Lu
e-mail: y.lu@ieee.org

1 Introduction

To make a manufacturing enterprise more competitive and profitable in the global marketplace, the profit driven "make-to-order" or "make-to-stock" business model has been popularly applied in manufacturing. Among multi-dimensional business and production decisions, computer-aided production scheduling to optimize desired objective criteria subject to multiple sophisticated constraints has been one of the most important decisions in business and production intelligence. In general, many production scheduling can be formulated as a constrained combinatorial optimization model. Eventually, it is a typical NP-hard problem, particularly for those large scale real-world applications. This kind of scheduling problems is difficult to be solved with traditional and exact optimization techniques. Consequently, many approximation methods, e.g., meta-heuristics, have been the major approach to this kind of constrained combinatorial optimization problems. Although approximation algorithms do not guarantee achieving optimal solutions, they can attain near-optimal solutions with reasonable computation time.

The major contributions to this paper are summarized as follows:

(1) Based on a class of production scheduling problems in manufacturing, this paper presents a mathematical model with an objective function, which can be computationally evaluated during the optimizing process.

(2) A hybrid evolutionary algorithm, in which extremal optimization (EO) plays the role of mutation operator in genetic algorithm (GA), is developed to solve the NP-hard constrained combinatorial optimization problem.
(3) The proposed evolutionary algorithm is applied and tested in solving a real-world production scheduling problem, i.e.,, the hot rolling scheduling in steel industry, with production scale data.

The rest of this paper is organized as follows: The scheduling problem is defined and formulated in Sect. 2. Section 3 presents the conceptual introduction to GA and EO, and proposes the hybrid evolutionary methodology for solving the production scheduling problem. Section 4 provides experimental results and performance analysis by simulating production scale data of hot rolling scheduling problem. Finally, Sect. 5 presents some concluding remarks of this paper.

2 Problem formulation

This section seeks to formulate the proposed scheduling problem in a mathematical model.

2.1 Problem description

The scheduling problem studied in this paper primarily arises from hot rolling production scheduling [1]. Concisely stated, the manufacturing systems under study have the following major features:

(1) The manufacturing processes could be either discrete, batch or semi-batch production.
(2) Without loss of generality, we consider here a manufacturing system with a single production stage and huge number of products described by multiple entities, such as product grades, dimensions and many other feature parameters.
(3) The number of work orders to be scheduled is equal to or greater than the number of scheduled orders within the resulting schedule scenario. This means that we may leave some orders unscheduled under a given time horizon.
(4) The objective is to construct an optimal schedule scenario that minimizes the weighted sum of sequence-dependent transition costs, non-execution penalties and earliness/tardiness (E/T) penalties.
(5) Multiple constraints, such as the feasible path between two consecutive orders, the capacity requirement of scheduling solution, due delivery date, etc., are taken into consideration in practice.

In those scheduling systems, there are two representative characteristics: (i) selecting a subset of orders from manufacturing orders to be processed; (ii) determining the optimal production sequence under practical constraints.

For scheduling of daily operations of a steel rolling, Balas and Martin (1985) firstly presented the prize collecting TSP (PCTSP) Model [2, 3]. Generally, the PCTSP can be defined as follows: A weighted graph $G = (V, A)$ is given, where $V = \{v_1,..., v_n\}$ is the set of n vertices and A is the set of directed arcs, a prize p_i and a penalty γ_i are associated with each vertex $i \in V$, and a cost c_{ij} is associated with each arc $(i, j) \in A$. The objective is to minimize the sum of costs between pairs of vertex and penalties for those unvisited vertex, while collecting a prescribed amount of prize money. Considering a special case when all the prizes are equal, Balas [1] presented the uniform PCTSP, in which $\sum_{i \in V} y_i = m, (m \leq n)$, where y_i is a 0–1 variable taking the value 1 if the vertex is visited and 0 if the vertex is not visited. Lopez et al. [4] proposed a detailed description of hot rolling scheduling and developed a fast and effective heuristic based on tabu search to solve the production scheduling problem. Tang and Wang [5] presented a new variant of the vehicle routing problem (VRP), i.e.,, the prize-collecting VRP, to formulate the similar production scheduling problem, and proposed an iterated local search algorithm using cyclic transfer. Furthermore, the scheduling problem with sequence-dependent transition effects and its variations, non-execution penalties, E/T penalties has also drawn considerable attention in manufacturing systems, because any reduction directly translates into production cost savings. Feo et al. [6] studied a single scheduling problem with sequence dependent setup costs and linear delay penalties, and then proposed a greedy randomized adaptive search procedure to solve the relevant problem. Shin et al. [7], and Pugazhendhi et al. [8] studied the scheduling problem with sequence-dependent set-up times. Rubin and Ragatz [9] applied a genetic algorithm to the sequencing problem with sequence dependent setup times for minimizing the total tardiness of a set of jobs in a single stage process. Ruben et al. [10], Liu and Chang [11] investigated the multi-stage scheduling problem with sequence dependent effects. Refael and Mati [12] studied the scheduling problem with non-execution penalties, i.e., each non-executed order is penalized.

Summarizing the above-mentioned literatures, the scheduling problem under study in this paper is a constrained combinatorial optimization problem, and unfortunately belongs to the class of NP-hard problems.

2.2 Mathematical formulation

Let us assume that there are n orders, and all orders are available for sequentially processing at the initial time of the defined schedule horizon. Consequently, the constrained combinatorial scheduling problem can be formulated as follows: Let $N = \{1,2,..., n\}$ be the set of manufacturing orders to be processed. The sequence-dependent transition cost c_{ij} is incurred while processing order j immediately after order i for any pairs of orders $i, j \in N$. We assign a comparatively large number to c_{ij} while it is infeasible from order i to j. The objective is to find out the optimal production sequence $S^* = \{j_1, j_2, \ldots, j_m\} (m \leq n)$ in the feasible solution space.

Based on above discussions, a mathematical model can be formulated. Let us first define the following notations:

i,j	index of orders
n	number of manufacturing orders in order book
m	number of orders in scheduling sequence
c_{ij}	sequence-dependent transition cost for processing order j immediately after order i
c_{ii}	non-execution penalty for order i isn't included in scheduling sequence
d_i	due delivery or complete time for order i
p_i	processing time of order i
t_i	starting time of order i
T_s	starting time of scheduling horizon

We also define the following decision variables:

$$x_{ij} = \begin{cases} 1, & \text{if order } j \text{ is preceded by order } i \\ 0, & \text{otherwise} \end{cases} \quad i, j \in N$$

Because we intend to solve the selection and sequencing operation simultaneously, another variable x_{ii} is defined as below.

$$x_{ii} = \begin{cases} 1, & \text{if order } i \text{ isn't included in scheduling sequence} \\ 0, & \text{otherwise} \end{cases} \quad i \in N$$

In addition, a dummy order 0 which has no processing time and sequence-dependent transition costs with other orders is added to the set of manufacturing orders as the starting node, and adjunctive constraints force the dummy to be included in scheduling sequence.

The scheduling solution is defined as an optimized production sequence with starting times calculated for each selected order. The objective function for evaluating the performance of schedule scenarios consists of two parts: (i) sequence-dependent transition costs and non-execution penalties, which can be formulated as $\sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ij}$; (ii) E/T penalties, in which early and tardy orders are penalized based on just-in-time philosophy [12]. Let a nonnegative earliness penalty weight α_i and a nonnegative tardiness penalty weight β_i be associated with each order $i \in N$. So the E/T penalties of total manufacturing orders can be defined as $\sum_{i=1}^{n} (\alpha_i e_i + \beta_i r_i)$, where $e_i = \max\{0, d_i - t_i - p_i\}$, $r_i = \max\{0, t_i + p_i - d_i\}$. An important special case being considered is $\alpha_i = \beta_i = 1$ for $1 \leq i \leq n$, accordingly, the E/T penalties are equal to $\sum_{i=1}^{n} (e_i + r_i)$. Consequently, the mathematical model of the proposed scheduling problem can be formulated as follows:

Minimize

$$\lambda \sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ij} + (1-\lambda)\sum_{i=1}^{n}(e_i + r_i) \tag{1}$$

Subject to

$$\sum_{i=0}^{n} x_{ij} = 1, \quad j = 0, \ldots n \tag{2}$$

$$\sum_{j=0}^{n} x_{ij} = 1, \quad i = 0, \ldots n \tag{3}$$

$$\sum_{i=1}^{n} (1 - x_{ii}) = m, \quad i = 1, \ldots n \tag{4}$$

$$x_{00} = 0 \tag{5}$$

$$t_0 = T_s \tag{6}$$

$$t_j = \sum_{i=0}^{n} x_{ij}(t_i + p_i) + x_{jj}(t_0 + T), \quad j = 1, \ldots n \tag{7}$$

$$x_{ij} \in \{0,\ 1\}, \quad i,j = 0, \ldots n \tag{8}$$

Equation (1) is the objective function of the scheduling problem, and the parameter $\lambda (0 \leq \lambda \leq 1)$ is the weight for adjusting the relative importance of two parts; constraint Eqs. (2) and (3) denote that each order is scheduled only once or not included in scheduling sequence respectively; Eq. (4) defines the capacity requirements; constraints (5) and (6) specify the dummy order as the starting node of scheduling sequence; Eq. (7) establishes the relationship between variables t_j and x_{ij}, in which constant T is greater than the processing time of one production sequence. The expression infers that if the order j is not included in current scheduling sequence, its starting time will be postponed to next scheduling sequence.

2.3 Complexity of solutions

Obviously, the proposed scheduling problem is a hard constrained combinatorial optimization problem with

$p_n^m = n!/m!$ possible solutions. Considering a simple case, we see 10 manufacturing orders are to be processed and seven orders are required to construct a production sequence. There will be 720 (10!/7!) possible solutions without considering any constraints. If we have a large amount of candidates, e.g., thousands of manufacturing orders in order book, the complete enumeration of all possible solutions is computationally prohibitive, i.e.,, no exact algorithm is capable of solving the optimization problem with reasonable computation time. Frequently, evolutionary algorithms as promising approximate techniques, such as genetic algorithm [10], tabu search [4], extremal optimization [13], are employed to solve the production scheduling problem for finding desirable, although not necessary optimal solution.

3 Hybrid evolutionary solution with marriage of GA and EO

In this section, we illustrate the detailed development of the hybrid evolutionary algorithm in solving the proposed scheduling problem.

3.1 Genetic algorithm

Genetic algorithm, which is enlightened by Darwinian evolutionary theory [14], can be applied to solve combinatorial optimization problems. By simulating the natural behaviour of biological systems, the "fittest" chromosomes (i.e.,, solutions) with desirable evaluation values are reproduced by the genetic operators, i.e., selection, crossover and mutation. Eventually, the objective function value of some individuals among the solution population possibly approaches to the global optimum. GA is a class of population-based search techniques, and the iterative improvement is realized by the generate-test [15] procedure of multi-solutions. It has been proved that GA is a powerful optimization technique and especially adaptive for practical applications.

3.2 Extremal optimization

Extremal optimization was recently proposed by Boettcher and Percus [13, 16]. It is inspired by self-organized critical models of co-evolution abstracted from the fundamental of ecosystem. The search process of EO eliminates those components having extremely undesirable (worst) performance in sub-optimal solution, and replaces them with randomly selected new components iteratively. Finally, the high-quality solutions of hard optimization problems may be explored through such kind of local searches.

Firstly, we interpret the novel algorithm by minimization problems. The basic algorithm operates on a single solution S, which usually consists of a number of variables $x_i (1 \le i \le n)$. At each update step, the variable x_i with worst fitness λ_i (individual cost contribution) is identified to alter.

The algorithm of EO can be represented as follows [17]:

```
Step 1:  Initialize a configuration $S$; set $S_{best} = S$.
Step 2:  For the "current" configuration $S$
    (a)  evaluate $\lambda_i$ for each variable $x_i$,
    (b)  find $j$ with $\lambda_j \ge \lambda_i$ for all $i$, i.e., $x_j$ has the worst fitness,
    (c)  choose at random a $S' \in N(S)$ such that the "worst" $x_j$ must change its state,
    (d)  if $C(S') < C(S_{best})$, then set $S_{best} = S'$,
    (e)  accept $S \leftarrow S'$ unconditionally, independent of $C(S')$ - $C(S)$.
Step 3:  Repeat at step (2) as long as desired.
Step 4:  Return $S_{best}$ and $C(S_{best})$.
```

There are no parameters to be adjusted for the selection of better solutions. To improve the results and avoid the possible dead ends, Boettcher and Percus [16] subsequently proposed τ-EO that is regarded as a general modification of EO by introducing a parameter. All variables x_i are ranked according to the relevant fitness λ_i, namely find a permutation Π: $\lambda_{\Pi(1)} \ge \lambda_{\Pi(2)} \ge \ldots \ge \lambda_{\Pi(n)} 1$. Then, each independent variable x_i to be moved is selected according to the probability distribution $P_k \propto k^{-\tau}$, $1 \le k \le n$ which is associated with the distinct ranks k_1, $k_2 \cdots k_n$. The choice of power law distribution ensures that no ranks get excluded for further evolution while maintaining a bias against variables with bad fitness.

EO and its derivatives have been extensively applied to solve numerous NP-hard combinatorial optimization problems. The simulation performance has been proved that EO outperforms other state-of-the-art algorithms in many applications, such as graph bi-partitioning, satisfiability (*MAX-K-SAT*), TSP problems and some industrial applications [13, 16–18].

3.3 Hybrid evolutionary methodology

GA makes a population-based evolutionary search on entire search space. It can explore the entire gene pool of solution configurations in which the crossover operation performs global exchanges and the mutation operation enhances the diversity of the population. Contrarily, EO exploits a single solution, with improvements achieved by repeatedly eliminating those components producing the worst fitness. It performs a local search that does not get stuck in the local minima, but proceeds to explore near-optimal configurations broadly. By combining the population-based search capacity of GA and the fine-grained local search efficacy of EO, we developed a hybrid evolutionary algorithm in this paper. It is credible to be an effective approach for solving constrained combinatorial optimization problems. In the following section, we will introduce the details of the hybrid evolutionary algorithm through the proposed scheduling problem.

3.3.1 Allocating & sequencing algorithm – modified GA for global search

Genetic algorithm and its derivatives have been widely applied to deal with combinatorial optimization problems. In this paper, a modified GA (MGA) is developed to solve the scheduling problem.

(1) Representation of solutions (or chromosomes)

The traditional GA applications use binary strings or ordinal integers to represent the chromosomes of solutions. Unlike the standard GA configuration, we define the scheduling solution as a chromosome that consists of the chain of genes in this application. The chromosome and genes represent the production sequence, and the sequenced orders marked with order ID respectively. Assuming that the length of the chromosome is equal to m, and n is the total amount of manufacturing orders, we see that the objective of the scheduling system is to build a scheduling solution with sequential orders selected from the given work orders to minimize the predefined criteria subject to multiple constraints.

For example, the representative of a chromosome can be shown in Fig. 1. The vector [5, 9, ... , 10] denotes the production sequence, and each number in the vector represents a particular order ID.

This chromosome can represent one possible scheduling solution in which the production sequence consists of 8 orders while total amount of candidates n=12.

(2) Population initialization

Generally, there are two issues to be considered for population initialization of GAs: the initial population size and the procedure to initialize the population [19].

For improving the capacity of searching solution space, the population size may increase exponentially with the complexity of the problem, i.e.,, the length of the chromosome. Nevertheless, a large population size is faced with excessive time complexity and high computational cost; on the contrary, a small quantity of individuals cannot efficiently locate the optimal solution. So, determining an appropriate population size is crucial to find the optimal scheduling solution. The population size is set as a default number Pop=200 in this application.

Furthermore, heuristic initialization and random initialization are among the most popular ways to generate the initial population. Heuristic methods can create sub-optimal solutions with high mean fitness. It is advantageous to improve the convergent speed, and yet, it may get into local optimum and never explore the whole solution space due to lacking of genetic diversity. Contrarily, if the initial population is generated in a random manner, it may take more generations to refresh and improve those defective individuals. Especially for a practical application with many constraints, starting from a randomly initialized population is difficult to search the optimal solution. Consequently, three well designed methods are employed to initialize the population in MGA. Firstly, a heuristic algorithm is applied to generate a feasible individual, which can improve the feasibility of the iterative-generated solutions in each generation, and accelerate the convergent speed. Secondly, the nearest neighbour search method, that firstly specifies a starting order i, and then iteratively visits next unscheduled order with least cost, is adopted to initialize partial individuals. Finally, random insertion method is used to create other individuals.

(3) Fitness function

The fitness function calculates how fit an individual is, and the "fittest" ones have more chances to be inherited into next generation. The scheduling solution S is evaluated by the objective function (1) of the mathematical model formulated in Sect. 2.2.

(4) Genetic operators

A. Selection

The selection operator is intended to improve the mean evaluation value of the population by giving the better chromosomes with higher probability to get evolved into next generation.

The selection schemes are characterized as the selection pressure, which is defined as the ratio of the selection probability of the best chromosome to that of an average chromosome. The rank-based selection schemes select individuals according to their fitness. In MGA, the formulation of selection probability proposed by Michalewicz [14] is applied: $p_i = c(1\text{-}c)^{i-1}$, where c is the selection pressure, and i is the sequence number of the ranked chromosomes within the population.

Selection operation used in MGA determines which individuals will have their genotypic information passed to next generation. One parent is chosen from the solution pool by roulette wheel selection. For maintaining the genetic diversity, another parent is selected from current generation using a method inspired by the niche technique, which ensures the differentia between two parent chromosomes. So, we define a similarity coefficient $c_{ij} = s_{ij}/n$, where n is the chromosome size, s_{ij} is the amount of

Slot#:	1	2	3	4	5	6	7	8
Chromosome:	5	9	3	12	7	11	2	10

Fig. 1 Example of representation and solution scheme

identical genes between chromosome i and chromosome j. Those chromosomes, with which the similarity coefficient of the first selected parent is less than a predetermined number c_0, have the potential to be chosen as the second parent. This method is proved to be effective to extensively explore the solution space.

B. Crossover

Crossover transforms the current solutions for finding better ones. Ordered crossover (OX) and partially matched crossover (PMX) [20] have been proved as the effective crossover schemes for integral encodes. Because the integral range of genes is equal to or greater than the chromosome length, i.e., the species of child chromosomes are not completely homologous to that of parents, the mechanism of the MGA crossover is not the same as that of the standard OX and PMX.

C. Local search as mutation

The population undergoes mutation by an actual change of the candidate chromosomes. Physically, it generates an alternative solution from a specific individual. In MGA, local search 2-*opt* are employed as the mutation operator.

While the terminated criteria are satisfied, the algorithm reports the best schedule scenario so far. The criteria of termination can be a certain number of generations (Gen), or a predefined amount of CPU time.

Here, it is worth noting that all algorithms were coded in C++, compiled by *MS Visual Studio 6.0*, and run on *Pentium 2.4 GHz CPU*. By simulating a typical group of production scale data gathered from hot rolling mill, the convergent curve of MGA is shown in Fig. 2. It is obvious that the proposed MGA (crossover probability p_c=0.95, mutation probability p_m=0.05) can converge at a near-optimal solution within a hundred of generations.

3.3.2 Local improving algorithm - τ-EO

Considering the characteristics of the proposed scheduling problem, we find that the localized sequence-dependent transition costs play an important role in making the resulting schedule solution satisfactory. So, we develop a novel local improving algorithm - τ-EO to exploit a specific scheduling solution. By introducing the parameter τ as analysed in Sect. 3.2, this fine-grained local search algorithm can explore the solution space broadly. Firstly, the localized fitness of extremal optimization is defined as $\lambda_i = c_{p(i)i} + c_{is(i)}$, where $p(i)$ and $s(i)$, respectively, represent the predecessor and successor of order i in scheduling sequence, i.e.,, the fitness

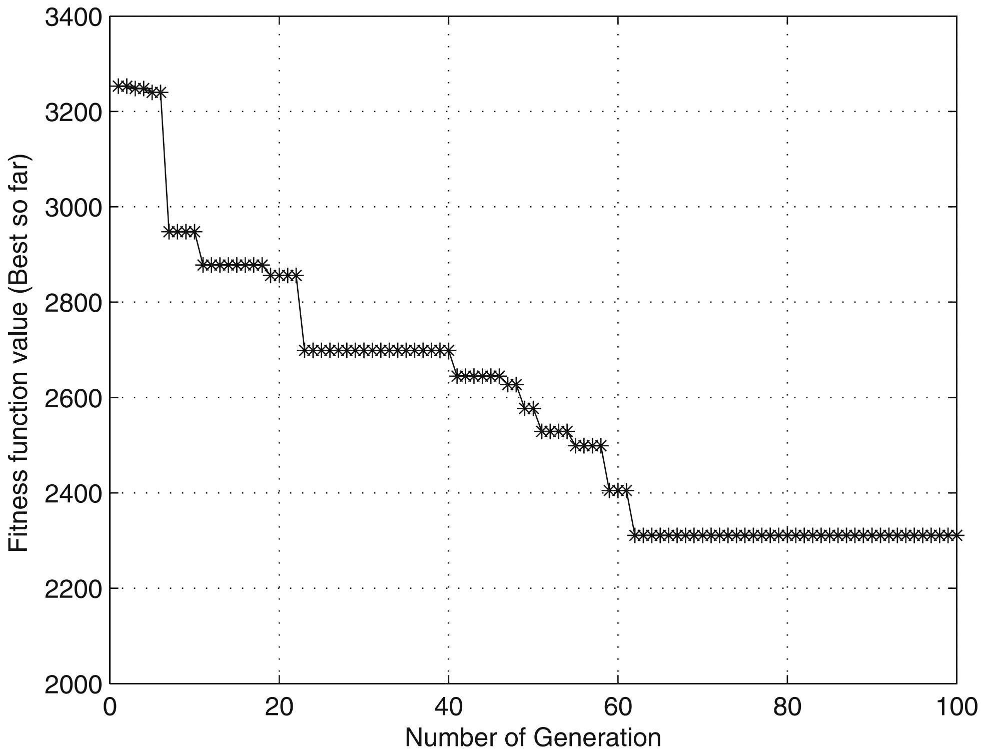


Fig. 2 Convergent curve of the modified genetic algorithm

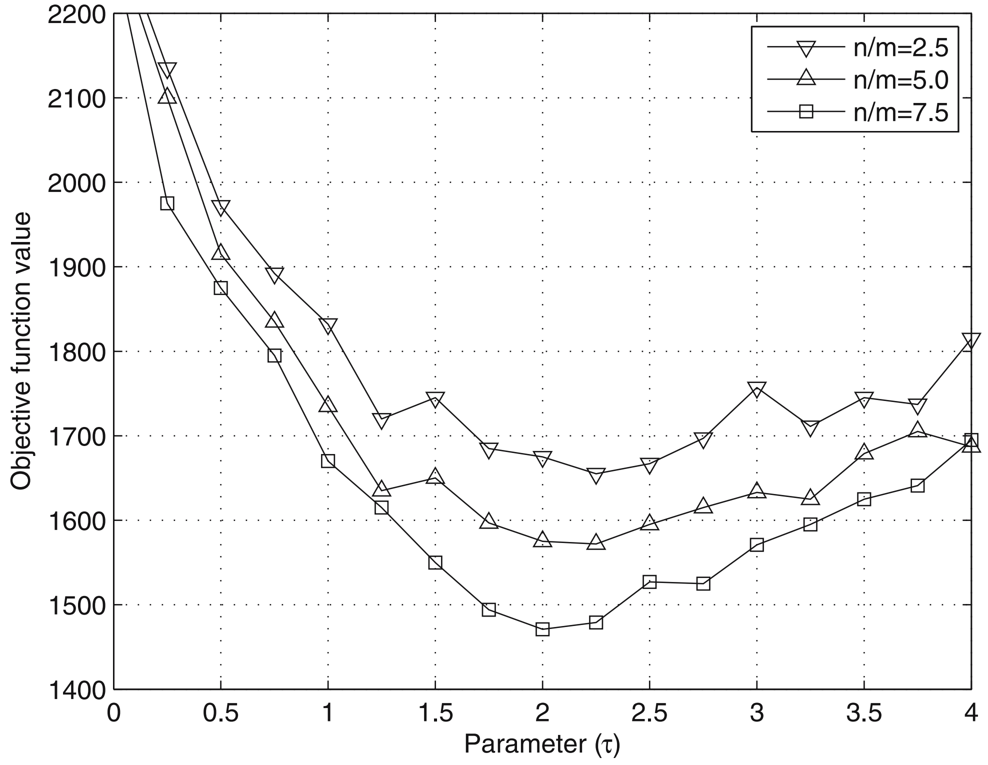


Fig. 3 Plot of objective function value over the parameter τ under a number of scheduling instances

of scheduled order i is the sum of two related transition costs. The workflow of τ-EO can be written as follows:

Step 1 Initialize parameters and obtain the initial solution S, which can be inherited from other algorithms, and then calculate the objective function $F(S)$, set $S_{best} = S$.

Step 2 For current scheduling solution S, sequentially evaluate the localized fitness λ_i for each scheduled order in the resulting schedule sequence, and rank them according to their fitness values.

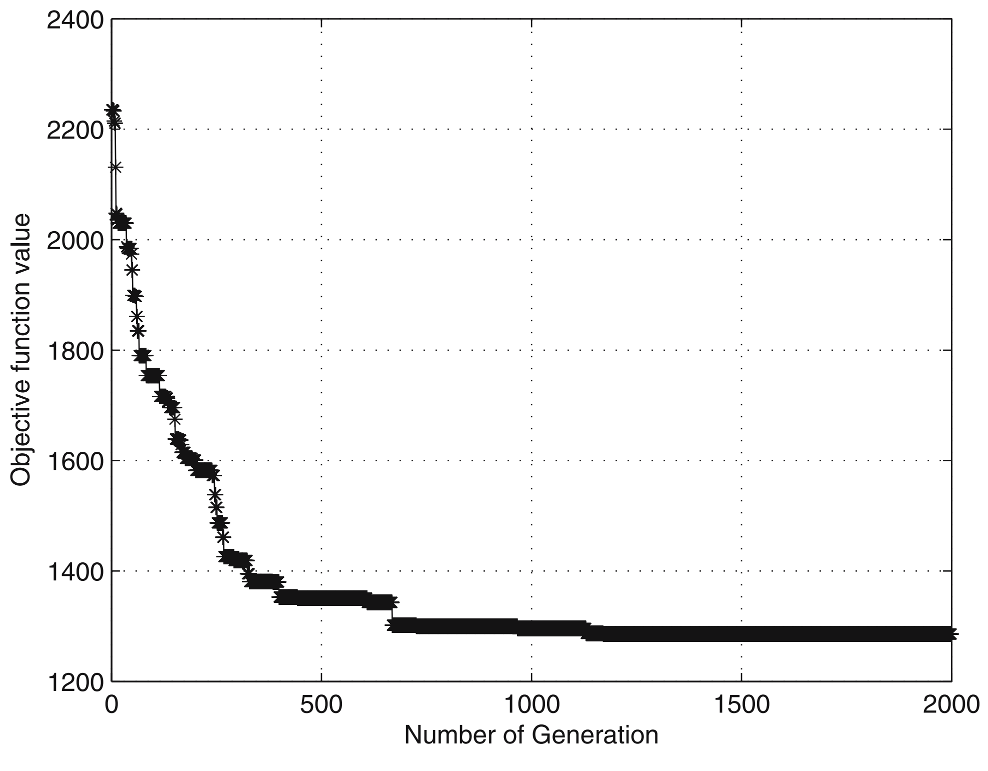
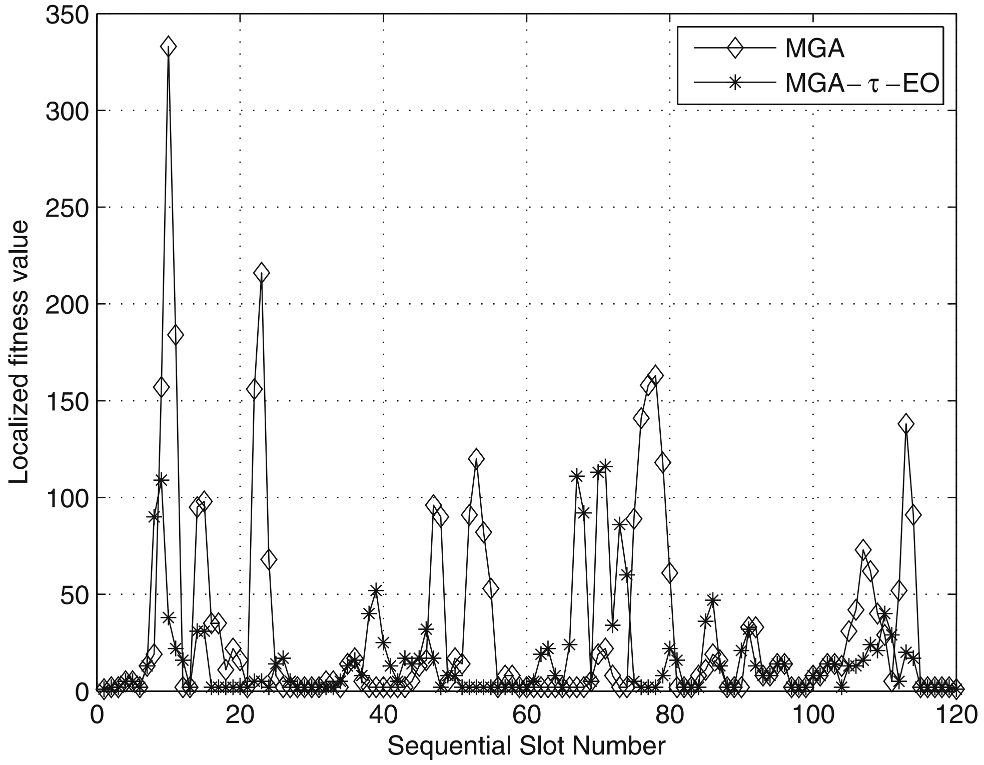


Fig. 4 Convergent curve of the proposed τ_EO algorithm

Fig. 5 Localized fitness comparison between MGA and MGA _ τ_EO



Step 3 Select an order $o(s)$ according to the power law distribution p_k (τ_0)*(*τ_0 *is a given value of the adjustable parameter).*

Step 4 Choose the best solution S' from a neighbor subspace $N(S)$ of the schedule solution S.

Step 5 Set $S_{best} = S'$ while $F(S') < F(S_{best})$, accept $S \leftarrow S'$ unconditionally.

Step 6 If the termination criteria aren't satisfied, go to step-2, else go next.

Step 7 Return S_{best} and $P(S_{best})$.

Note that in step 4 the neighbor $N(S)$ can be built by various strategies. The "route-improvement" algorithm being similar to the heuristic search proposed by Cowling [21] is used to generate the solution subspace. It takes the scheduling solution S and improves it by perturbing S slightly. This perturbation is iterated until no further improvement is possible, and then the local optimum S' is obtained. The applied heuristics are:

Delete the selected order $o(s)$ from the scheduling sequence S;

Select an unscheduled order $o(u)$, and insert it into the optimum position;

Accept the feasible transitional solution as an element of $N(S)$ if $\lambda_o(u) < \lambda_o(s)$;

Repeat steps 2 and 3 until all unscheduled orders have performed the procedures.

Three types of improving moves corresponding to the adjusting rules usually are employed by manual schedulers. Hence the local optimum is quite practical in real-world manufacturing systems.

Because τ-EO has only one adjustable parameter τ, we analyze its optimal choice through simulating a serial of parameter values under an identical solution provided by MGA. The asymptotic curve of objective function value over the adjustable parameter has been verified in the practical scheduling problem as exemplified in Fig. 3.

Generally speaking, the algorithm reaches an optimal solution at a prediction value $\tau_{opt} \approx 2.0$, and the objective function value rises sharply beyond the neighbor of τ_{opt}, especially for the scheduling instances with large n/m.

Figure 4 shows the convergent curve of τ-EO algorithm. The initial solution is inherited from previous MGA. It considerably improves the schedule solution and reaches a convergent solution within 2000 generations. The procedure takes about 45 seconds while n=1050, and m=120.

3.3.3 Hybrid evolutionary algorithms

Straightforwardly, the scheduling scenario given by MGA can be further optimized through τ-EO as demonstrated above. For exploiting the respective advantages of GA and EO further, a hybrid evolutionary algorithm with marriage of GA and EO, which is called a genetic extremal optimization (GEO) in the following parts, is presented. In the integrated method, τ-EO plays the role of the mutation operator of MGA. So, its workflow is mainly consistent

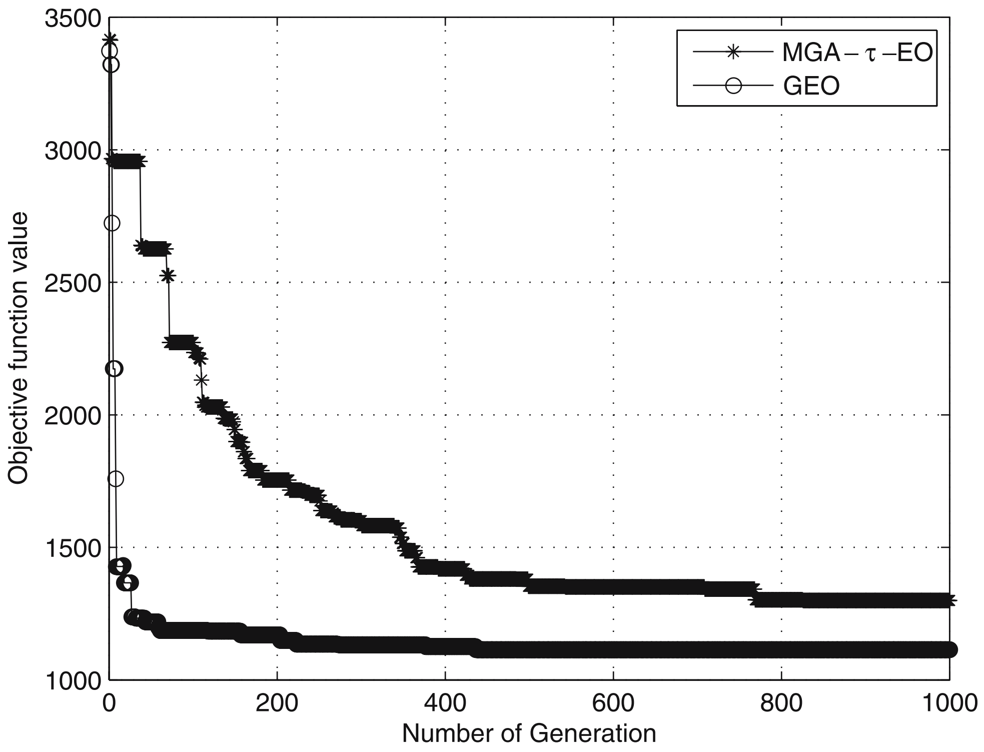


Fig. 6 Evolutionary process comparison between MGA _ τ_EO and GEO

with MGA as described above, except that the mutation operation is replaced by τ-EO.

It has been proved that this method may provide even superior scheduling solution by experimental results. Now we summarize and compare the test results for practical scheduling data.

Figure 5 shows the comparisons in the localized fitness between MGA and MGA - τ-EO, in which the scheduling solution of MGA is further optimized by τ-EO. It is obvious that τ-EO can improve the scheduling solution of MGA through consecutively replacing those undesirable or underperformed orders in scheduling sequence.

In the proposed hybrid evolutionary algorithm GEO, $Pop \times Gen \times p_m$ possible solutions will be optimized by τ-EO. Using same production data, the comparisons in evolutionary processes given by MGA - τ-EO and GEO are illustrated in Fig. 6.

Evidently, GEO provides superior performances in both scheduling quality and convergent speed compared with GA - τ-EO.

Table 1 Improvement created by the hybrid evolutionary algorithm using practical production scale data

Instance No.	n/m	MGA	MGA- τ-EO		GEO	
		Objective	Objective	Improvement	Objective	Improvement
1	2.5	2404	2237	6.95	2035	15.35
2	2.5	3312	2913	12.05	2610	21.20
3	2.5	2453	2337	4.73	1941	20.87
4	2.5	2855	2552	10.61	2530	11.38
5	5	1981	1509	23.83	1472	25.69
6	5	1889	1596	15.51	1286	31.92
7	5	1952	1613	17.37	1523	21.98
8	5	1794	1399	22.02	1302	27.42
9	7.5	1715	1325	22.74	1147	33.12
10	7.5	1479	1202	18.73	1074	27.38
11	7.5	1637	1237	24.43	1122	31.46
12	7.5	1529	1219	20.27	1035	32.31

4 Experimental results and performance analysis

Extensive experiment tests have been performed with production scale data. Some comparison results of objective function values for MGA, MGA - τ-EO, and GEO are summarized in Table 1.

It can be seen that the hybrid evolutionary algorithm developed in this paper can further reduce the objective function value and provide a more favorable scheduling solution. Especially, the significant effects of GEO are demonstrated by 30% reduction for those scheduling instances with larger n/m.

5 Concluding remarks

In this paper, we studied a typical constrained combinatorial scheduling problem with characteristics of sequence dependent transition cost, non-execution penalties and E/T penalties. A mathematical model is proposed to formulate it. Since this problem is hardly possible to be solved with exact optimization methods, an effective hybrid evolutionary algorithm with marriage of GA and EO is developed to solve the proposed scheduling problem. The experimental results with production scale data provide the quantitative comparisons in scheduling performance and computation efficiency. We can see that the GEO using τ-EO as the mutation operator of MGA provides superior performance. Based on this research, it is intuitively clear that the state-of-the-art approach has great potentials in both academic research and industrial applications.

Acknowledgments This research is supported by National Natural Science Foundation of China (Grant No: 60574063).

References

1. Balas E (1999) New classes of efficiently solvable generalized Traveling Salesman Problems. Ann Oper Res 86:529–558
2. Balas E, Martin G (1985) Roll-a-Round: Software package for scheduling the rounds of a rolling mill. Balas and Martin Associates, Pittsburgh
3. Feillet D, Dejax P, Gendreau M (2005) Traveling salesman problems with profits. Transport Sci 39(2):188–205
4. Lopez L, Carter MW, Gendreau M (1998) The hot strip mill production scheduling problem: A tabu search approach. Eur J Oper Res 106:317–335
5. Tang LX, Wang XP (2006) Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem. Int J Adv Manuf Technol 29(11–12):1246–1258
6. Feo TA, Sarathy K, McGahan J (1996) A GRASP for single machine scheduling with sequence dependent setup costs and linear delay penalties. Comput Oper Res 23(9):881–895
7. Shin HJ, Kim CO, Kim SS (2002) A tabu search algorithm for single machine scheduling with release times, due dates, and sequence-dependent set-up times. Int J Adv Manuf Technol 19:859–866
8. Pugazhendhi S, Thiagarajan S, Rajendran C, Anantharaman N (2004) Generating non-permutation schedules in flowline-based manufacturing systems with sequence-dependent setup times of jobs: a heuristic approach. Int J Adv Manuf Technol 23:64–78
9. Rubin PA, Ragatz GL (1995) Scheduling in a sequence dependent setup environment with genetic search. Comput Oper Res 22(1): 85–99
10. Ruben R, Concepcion M, Javier A (2005) Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. Eur J Oper Res 165:34–54
11. Liu CY, Chang SC (2000) Scheduling flexible flow shops with sequence-dependent setup effects. IEEE Trans Robot Autom 16(4): 408–419
12. Refael H, Mati S (2005) Machine scheduling with earliness, tardiness and non-execution penalties. Comput Oper Res 32:683–705
13. Boettcher S, Percus AG (1999) Extremal optimization: Methods derived from Co-Evolution. In: Proceedings of the genetic and evolutionary computation conference, pp 825–832
14. Michalewicz Z (1996) Genetic algorithms + data structures = evolution programs. Springer-Verlag, London, UK
15. Han J (2005) Local evaluation functions and global evaluation functions for computational evolution. Complex Systems 05:1–41 http://dx.doi.org/SFI-WP 03-09-048
16. Boettcher S, Percus AG (2000) Nature's way of optimizing. Artif Intell 119:275–286
17. Boettcher S, Percus AG (2003) Optimization with extremal dynamics. Complexity (Wiley Periodicals, Inc.) 8(2):57–62
18. De Sousa FL, Vlassov V, Ramos FM (2004) Generalized extremal optimization: An application in heat pipe design. Appl Math Model 28:911–931
19. Chang WA, Ramakrishna RS (2002) A genetic algorithm for shortest path routing problem and the sizing of population. IEEE Trans Evol Comput 6(6):566–579
20. Larraòaga P, Kuijpers CMH, Murga RH, Inza I, Dizdarevic S (1999) Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artif Intell Rev 13:129–170
21. Cowling P (2003) A flexible decision support system for steel hot rolling mill scheduling. Comput Ind Eng 45:307–321