Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2010, Article ID 781598, 14 pages
doi:10.1155/2010/781598

Research Article

Modeling and Evolutionary Optimization on Multilevel Production Scheduling: A Case Study

Ruifeng Shi,[1] Chunxia Shangguan,[2] and Hong Zhou[3]

[1] *School of Control & Computer Engineering, North China Electric Power University, Beijing 102206, China*
[2] *College of Engineering Science & Technology, Shanghai Ocean University, Shanghai 201306, China*
[3] *School of Economics & Management, Beihang University, Beijing 100083, China*

Correspondence should be addressed to Ruifeng Shi, shi.ruifeng@126.com

Received 31 August 2009; Revised 27 February 2010; Accepted 16 April 2010

Academic Editor: Chuan-Kang Ting

Multilevel production scheduling problem is a typical combinatorial optimization problem in a manufacturing system, which is traditionally modeled as several hierarchical sublevel problems and optimized at each level, respectively. An integrated model, which can cope with the whole multilevel scheduling information simultaneously, is proposed in this paper, and a specific evolutionary algorithm is designed to solve the integrated model with a twin-screw coding strategy. In order to evaluate the performance of the new algorithm, a real 3-level production scheduling problem is employed for case study, and two typical metaheuristic algorithms, a genetic algorithm (GA) and a simulated annealing (SA), are also employed for comparison study. Experimental simulation results show that our proposed modeling and optimization method has outperformed the other ones.

1. Introduction

Multi-level production scheduling problem is a typical combinatorial optimization problem in manufacturing system, which is traditionally modeled as several hierarchical sublevel problems, and solved by single-level scheduling methods level by level [1–4]. As we know, the trends of manufacturing system development are increasing the competitivity of the company, promoting the customers' service level, integrating the global manufacturing information, and processing flexible task flows in a more efficient way [5, 6]. In order to cope with these trends, researchers need to optimize plant operations and total activities from suppliers to customers and help manufacturers to find their ways in global optimization and support the needs of manufacturing at the same time. Along with all these scenarios, problem modeling and its optimization techniques play important roles in achieving these goals [7–10].

Generally speaking, multi-level production scheduling problem (MLPS) is one kind of hierarchical production planning problem, which considers a set of jobs on given machines with predefined sequence, while discarding to cope with a lot size problem [11–13]. Jobs in an MLPS problem belong to different product levels, and the higher level product cannot begin its process operation until all its subproducts in the lower level are finished. The general way to solve an MLPS problem is decomposing the entire problem into several sublevel problems according to its product level structure and optimize these subproblems within each level [14]. The product level structure of a typical MLPS problem can be illustrated as in Figure 1, in which job 1 belongs to level 1; jobs 2, 3 belong to level 2; jobs 4, 5, 6, 7, 8 belong to level 3. Besides, the process operation precedence between adjacent levels is as follows: job 2 (level 2) cannot begin its process operation, until all its sub-jobs 4, 5 (level 3) are finished; job 1 (level 1) cannot begin its process operation, until all its subjobs 2, 3 (level 2) are finished.

2. Related Works

During the past decades, some researches have tried their efforts to demonstrate that a hierarchical production planning technique is the most efficient way in solving a multistage, multilevel production planning problem [15–17], while other researchers believed in that a monolithic/integrated method is better than a hierarchical one.

These researchers proposed many integrated multi-level production planning models for various problems and obtained some exciting conclusions. But most of them are focused on coping a scheduling problem with a lot sizing problem simultaneously; few attention is taken on the MLPS described as in Figure 1. The reason may lied in the fact that a batch production module plays a major role in most manufacturing companies during the past decades, but a small batch, multiclass production module becomes more and more popular in current factories, especially in some high technological industries. This leads researchers to find more efficient models and optimization techniques to solve the MLPS in recent years [18–20].

Since the occurrence of modern heuristic optimization algorithms, like evolutionary algorithm (EA), simulated annealing (SA), coevolutionary algorithm (COEA), ant colony (AC), and particle swarm optimization (PSO), and so forth, solving a large scale MLPS with an integrated model becomes possible within an acceptable computational cost. More and more attentions are attracted to solve this NP complete (or NP-Hard) problem with an integrated model, and various problem-dependent heuristic algorithms are proposed to enhance the efficiency of the algorithms, among which evolutionary algorithm is the most attractive/preferred one [21–23].

Usually, an evolutionary algorithm is designed to solve a single-level scheduling problem, and an MLPS integrated model can be solved with a hierarchical looped EA, in which precedence prerequisite information can be satisfied with transferring the ready time and release time between adjacent levels. The result is greatly improved by contrast to a hierarchical technique. But the computational cost with this scheme may grow rapidly as the problem scale is increasing. Besides, the optimality of the final result is doubted by many classical optimization mathematicians. This leads researchers turning to new gene expression and evolving strategies to overcome those drawbacks.

In the following sections, Section 3 builds up a general integrate 3-level production scheduling optimization model. Section 4 proposes a twin-screw-coded hybrid evolutionary algorithm to solve the integrated model. Section 5 employs a real 3-level production scheduling case study to evaluate the performance of the proposed modeling and optimization technique, and Section 6 gives the conclusion of this study and highlights some perspectives for future study.

3. Mathematical Integrated Model for a Typical Three-level Production Scheduling Problem

In general, a multi-level production scheduling problem (Figure 1) can be described as follows: a final product consists of several (n) assemblies, each assembly consists of some ($n_{(ij)}(i = 1,2,\ldots,n)$) subassemblies, and each subassembly consists of several ($n_{(ij)}$ ($i = 1,2,\ldots,n$; $j = 1,2,\ldots,n_{(ij)}$)) parts,.... All the jobs, including assemblies, subassemblies, and parts, should be processed through $m_{(i)}$, $m_{(ij)}$, and $m_{(ijk)}$ operational sequences with given orders, where $m_{(i)}$ ($i = 1,2,\ldots,n$) represents the maximum operation number of

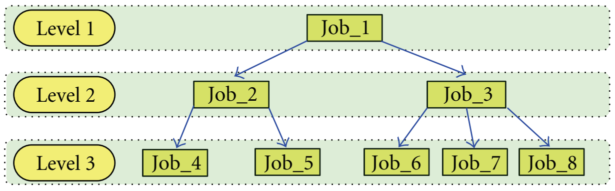


Figure 1: An example of hierarchical structure of MLPS problem.

assembly (i), $m_{(ij)}$ ($i = 1,2,\ldots,n$; $j = 1,2,\ldots,n_{(i)}$) represents the maximum operation number of subassembly (ij), and $m_{(ijk)}$ ($i = 1,2,\ldots,n$; $j = 1,2,\ldots,n_{(i)}$, $k = 1,2,\ldots,n_{(ij)}$) represents the maximum operation number of part (ijk). Besides, one assembly cannot begin its process operation until all of its subassemblies are finished and assembled into the assembly. The final product is assembled by the assemblies. Generally, $\mathbf{min}\{makespan\}$ is considered as the optimization objective. In this section, an integrated 3-level MLPS model is built up, which can be stated as (1)–(24):

$$\min \quad f = \min\{C_{\max}\} = \min\{\max\{C_{(i)}\}\} \tag{1}$$

$$\text{s.t.} \quad t^S_{[a](i)} \geq t^S_{[a](ij)} + p_{[a](ij)}; \quad a = 1,2,\ldots,m_{(i)}; \quad i = 1,2,\ldots,n; \quad j = 1,2,\ldots,n_{(i)}; \tag{2}$$

$$t^S_{[a](ij)} \geq t^S_{[a](ijk)} + p_{[a](ijk)}; \quad a = 1,2,\ldots,m_{(ij)}; \quad i = 1,2,\ldots,n; \quad j = 1,2,\ldots,n_{(i)}; \quad k = 1,2,\ldots,n_{(ij)}; \tag{3}$$

$$t^S_{[a](i)} + p_{[a](i)} \leq t^S_{[b](i)} + M\cdot(1 - q_{[a][b](i)}); \quad a,b = 1,2,\ldots,m_{(i)}; \quad i = 1,2,\ldots,n; \tag{4}$$

$$t^S_{[a](ij)} + p_{[a](ij)} \leq t^S_{[b](ij)} + M\cdot\left(1 - q_{[a][b](ij)}\right); \quad a,b = 1,2,\ldots,m_{(ij)}; \quad i = 1,2,\ldots,n; \quad j = 1,2,\ldots,n_{(i)}; \tag{5}$$

$$t^S_{[a](ijk)} + p_{[a](ijk)} \leq t^S_{[b](ijk)} + M\cdot\left(1 - q_{[a][b](ijk)}\right); \quad a,b = 1,2,\ldots,m_{(ijk)}; \quad i = 1,2,\ldots,n; \quad j = 1,2,\ldots,n_{(i)}; \quad k = 1,2,\ldots,n_{(ij)}; \tag{6}$$

$$t^S_{[g](i)} + p_{[g](i)} \leq t^S_{[g](j)} + M\cdot\left(1 - q_{[g]((i)(j))}\right); \quad g = 1,2,\ldots,m_{(i)}; \quad i,j = 1,2,\ldots,n; \tag{7}$$

$$t^S_{[g](ij)} + p_{[g](ij)} \leq t^S_{[g](ik)} + M\cdot\left(1 - q_{[g]((ij)(ik))}\right); \quad g = 1,2,\ldots,m_{(ij)}; \quad i = 1,2,\ldots,nj, \quad k = 1,2,\ldots,n_{(i)}; \tag{8}$$

$$t^S_{[g](ijk)} + p_{[g](ijk)} \le t^S_{[g](ijl)} + M \cdot \left(1 - q_{[g]((ijk)(ijl))}\right); \quad g = 1,2,\dots,m_{(ijk)}; \quad i = 1,2,\dots,n; \quad j = 1,2,\dots,n_{(i)}; \quad k,l = 1,2,\dots,n_{(ij)}; \tag{9}$$

$$t^S_{[g](i)} + p_{[g](i)} \le t^S_{[g](jk)} + M \cdot \left(1 - x_{[g]((i)(jk))}\right); \quad i,j = 1,2,\dots,n; \quad k = 1,2,\dots,n_{(i)}; \quad g \in \{1,2,\dots,m_{(i)}\} \cap \left\{1,2,\dots,m_{(jk)}\right\}; \tag{10}$$

$$t^S_{[g](jk)} + p_{[g](jk)} \le t^S_{[g](i)} + M \cdot \left(1 - x_{[g]((jk)(i))}\right); \quad i,j = 1,2,\dots,n; \quad k = 1,2,\dots,n_{(i)}; \quad g \in \{1,2,\dots,m_{(i)}\} \cap \left\{1,2,\dots,m_{(jk)}\right\}; \tag{11}$$

$$t^S_{[g](i)} + p_{[g](i)} \le t^S_{[g](jkl)} + M \cdot \left(1 - x_{[g]((i)(jkl))}\right); \quad i,j = 1,2,\dots,n; \quad k = 1,2,\dots,n_{(i)}; \quad l = 1,2,\dots,n_{(jk)}; \quad g \in \{1,2,\dots,m_{(i)}\} \cap \left\{1,2,\dots,m_{(jkl)}\right\}; \tag{12}$$

$$t^S_{[g](jkl)} + p_{[g](jkl)} \le t^S_{[g](i)} + M \cdot \left(1 - x_{[g]((jkl)(i))}\right); \quad i,j = 1,2,\dots,n; \quad k = 1,2,\dots,n_{(i)}; \quad l = 1,2,\dots,n_{(jk)}; \quad g \in \{1,2,\dots,m_{(i)}\} \cap \left\{1,2,\dots,m_{(jkl)}\right\}; \tag{13}$$

$$t^S_{[g](ij)} + p_{[g](ij)} \le t^S_{[g](kls)} + M \cdot \left(1 - x_{[g]((ij)(kls))}\right); \quad i,k = 1,2,\dots,n; \quad j,l = 1,2,\dots,n_{(i)}; \quad s = 1,2,\dots,n_{(ij)}; \quad g \in \left\{1,2,\dots,m_{(ij)}\right\} \cap \{1,2,\dots,m_{(kls)}\}; \tag{14}$$

$$t^S_{[g](kls)} + p_{[g](kls)} \le t^S_{[g](ij)} + M \cdot \left(1 - x_{[g]((kls)(ij))}\right); \quad i,k = 1,2,\dots,n; \quad j,l = 1,2,\dots,n_{(i)}; \quad s = 1,2,\dots,n_{(ij)}; \quad g \in \left\{1,2,\dots,m_{(ij)}\right\} \cap \{1,2,\dots,m_{(kls)}\}; \tag{15}$$

$$t^S_{[a](i)} \ge 0; \quad a = 1,2,\dots,m_{(i)}; \quad i = 1,2,\dots,n; \tag{16}$$

$$t^S_{[a](ij)} \ge 0; \quad a = 1,2,\dots,m_{(ij)}; \quad i = 1,2,\dots,n; \quad j = 1,2,\dots,n_{(i)}; \tag{17}$$

$$t^S_{[a](ijk)} \ge 0; \quad a = 1,2,\dots,m_{(ijk)}; \quad i = 1,2,\dots,n; \quad j = 1,2,\dots,n_{(i)}; \quad k = 1,2,\dots,n_{(ij)}; \tag{18}$$

$$x_{[g]((i)(j))} + x_{[g]((j)(i))} = 1, \quad \text{where } x_{[g]((i)(j))} = 0 \text{ or } 1; \quad g = 1,2,\dots,m_{(i)}; \quad i,j = 1,2,\dots,n; \tag{19}$$

$$x_{[g]((ij)(ik))} + x_{[g]((ik)(ij))} = 1, \quad \text{where } x_{[g]((ij)(ik))} = 0 \text{ or } 1; \quad g = 1,2,\dots,m_{(ij)}; \quad i = 1,2,\dots,n; \quad j,k = 1,2,\dots,n_{(i)}; \tag{20}$$

$$x_{[g]((ijk)(ijl))} + x_{[g]((ijl)(ijk))} = 1, \quad \text{where } x_{[g]((ijk)(ijl))} = 0 \text{ or } 1; \quad g = 1,2,\dots,m_{(ijk)}; \quad i = 1,2,\dots,n; \quad j = 1,2,\dots,n_{(i)}; \quad k,l = 1,2,\dots,n_{(ij)}; \tag{21}$$

$$x_{[g]((i)(jk))} + x_{[g]((jk)(i))} = 1 \quad (i) \ne (j), \quad \text{where } x_{[g]((i)(jk))} = 0 \text{ or } 1; \quad i,j = 1,2,\dots,n; \quad k = 1,2,\dots,n_{(i)}; \quad g \in \{1,2,\dots,m_{(i)}\} \cap \left\{1,2,\dots,m_{(jk)}\right\}; \tag{22}$$

$$x_{[g]((i)(jkl))} + x_{[g]((jkl)(i))} = 1 \quad (i) \ne (j), \quad \text{where } x_{[g]((i)(jkl))} = 0 \text{ or } 1; \quad i,j = 1,2,\dots,n; \quad k = 1,2,\dots,n_{(i)}; \quad l = 1,2,\dots,n_{(jk)}; \quad g \in \{1,2,\dots,m_{(i)}\} \cap \left\{1,2,\dots,m_{(jkl)}\right\}; \tag{23}$$

$$x_{[g]((ij)(kls))} + x_{[g]((kls)(ij))} = 1 \quad (ij) \ne (kl), \quad \text{where } x_{[g]((ij)(kls))} = 0 \text{ or } 1; \quad i,k = 1,2,\dots,n; \quad j,l = 1,2,\dots,n_{(i)}; \quad s = 1,2,\dots,n_{(ij)}; \quad g \in \left\{1,2,\dots,m_{(ij)}\right\} \cap \{1,2,\dots,m_{(kls)}\}. \tag{24}$$

The definition of the parameters in the model is as follows.

FS represents the feasible solution set; $o_{[a](i)}$ represents the ath operation of job assembly (i), $o_{[a](ij)}$ represents the ath operation of job subassembly (ij); $o_{[a](ijk)}$ represents the ath operation of job part (ijk); $p_{[a](i)}$ represents the operation time of $o_{[a](i)}$; $p_{[a](ij)}$ represents the operation time of $o_{[a](ij)}$; $p_{[a](ijk)}$ represents the operation time of $o_{[a](ijk)}$; M is an infinite positive multiply factor:

$$q_{[a][b](i)} = \begin{cases} 1, & \text{if the operation of assembly } (i) \text{ on machine } [b] \\ & \quad \text{is exactly after its operation on machine } [a], \\ 0, & \text{otherwise,} \end{cases}$$

$$q_{[a][b](ij)} = \begin{cases} 1, & \text{if the operation of subassembly } (ij) \text{ on} \\ & \quad \text{machine } [b] \text{ is exactly after its operation} \\ & \quad \text{on machine } [a], \\ 0, & \text{otherwise,} \end{cases}$$

$$q_{[a][b](ijk)} = \begin{cases} 1, & \text{if the operation of part } (ijk) \text{ on machine } [b] \\ & \quad \text{is exactly after its operation on machine } [a], \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

The definition of the decision variables in the model is as follows.

$t^S_{[a](i)}$ represents the start time of operation $o_{[a](i)}$, $t^S_{[a](ij)}$ represents the start time of operation $o_{[a](ij)}$, $t^S_{[a](ijk)}$ represents the start time of operation $o_{[a](ijk)}$ and $C_{(i)}$ represents the end time of assembly (i); thus we have

$$C_{(i)} = \max_{a \in 1,2,\ldots,m_{(i)}} \left(t^S_{[a](i)} + p_{[a](i)} \right), \tag{26}$$

and the 0-1 programming variables are defined as

$$x_{[g](i)(j)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{assembly } (i) \text{ on machine } [g] \text{ is } (j), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](ij)(ik)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{subassembly } (ij) \text{ on machine } [g] \text{ is } (ik), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](ijk)(ijl)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{part } (ijk) \text{ on machine } [g] \text{ is } (ijl), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](i)(jk)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{assembly } (i) \text{ on machine } [g] \text{ is} \\ & \quad \text{subassembly } (jk), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](jk)(i)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{subassembly } (jk) \text{ on machine } [g] \text{ is} \\ & \quad \text{assembly } (i), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](i)(jkl)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{assembly } (i) \text{ on machine } [g] \text{ is} \\ & \quad \text{part } (jkl), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](jkl)(i)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{part } (ijk) \text{ on machine } [g] \text{ is assembly } (i), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](ij)(kls)} = \begin{cases} 1, & \text{if the immediate successive operation of} \\ & \quad \text{subassembly } (ij) \text{ on machine } [g] \text{ is part } (kls), \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{[g](kls)(ij)} = \begin{cases} 1, & \text{if the immediate successive operation of part} \\ & \quad (kls) \text{ on machine } [g] \text{ is subassembly } (ij), \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

The physical explanation for the model (shown as (1)–(24)) is as follows.

Equation (1) gives the optimization objective as makespan; (2) constrains that the start time of an assembly must be later than the completion time of all its subassemblies; (3) constrains that the start time of a subassembly

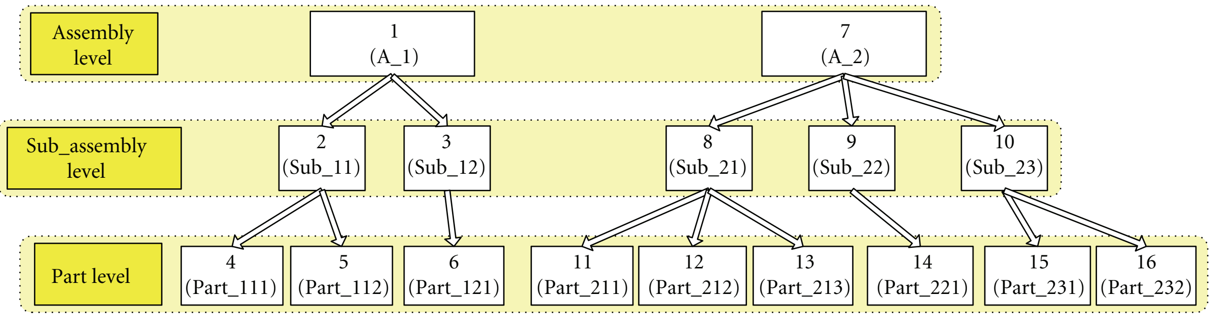


Figure 2: Example of a 3-level production scheduling problem.

must be later than the completion time of all its parts; (4) defines the precedence relationship of assembly (i)'s two successive operations $o_{[a](i)}$ and $o_{[b](i)}$; that is, the start time of operation $o_{[b](i)}$ must be later than the end time of operation $o_{[a](i)}$ if $q_{[a][b](i)} = 1$, and vice versa; (5) defines the precedence relationship between subassembly (ij)'s two successive operation $o_{[a](ij)}$ and $o_{[b](ij)}$; (6) defines the precedence relationship between part (ijk)'s two successive operation $o_{[a](ijk)}$ and $o_{[b](ijk)}$; (7)–(15) define that there can only process at most one job on machine $[g]$ at one time; (16)–(18) constrain the start time of each operation as a non-negative variable; (19)–(21) define the constraints between each related-pair of decision variables in each level to guarantee the feasibility of the solution; (22)–(24) define the constraints between each pair of decision variables in different levels.

4. Twin-Screw-Coded Evolutionary Algorithm for Multilevel Production Scheduling Optimization

Since the computational complexity of the multi-level production scheduling problem is very high, it is hard to solve it with current existing optimization methods efficiently (either the precise methods or the problem-dependent heuristic algorithms). In order to overcome the drawbacks of current existing methods, a twin-screw-coded evolutionary algorithm is proposed in this paper, which encodes a possible multi-level scheduling scheme in a twin-screw chromosome, and a metaheuristic-based population gap for elitist exchange and local search is employed to enhance the convergence of the algorithm.

A typical 3-level MLPS example is shown in Figure 2, which includes two assembles (1,7), five subassemblies (1's subassemblies include 2, 3; 7's subassemblies include 8, 9, 10) and nine parts (2's parts are 4, 5; 3's part is 6; 8's parts are 11, 12, 13; 9's part is 14; 10's parts are 15, 16). The hierarchical process precedence relationship between the jobs can be easily told from the figure, in which the process scheduling optimization covers all 3-levels' jobs at the same time.

4.1. Gene Expression: Twin-Screw Coding. Current existing evolutionary coding can only express one single level's scheduling information; it is hard to employ them to express a multi-level production scheduling information with one chromosome. Hence, we propose an operation-based twin-screw coding strategy to solve this problem: we define each job number with an implicit subsidiary coding (subcoding) to label its level information and construct a twin-screw module to express the hierarchy of the scheduling scheme. Assume that, in the example of Figure 2, each job in $\{3, 7, 12\}$ has two operations, respectively, job 5 has three operations, and all the other jobs have only one operation, which is a typical mixed flexible job shop problem. A feasible solution for this problem can be coded as a twin-screw gene (shown as Figure 3), in which the *italic* gene in the second row shows the level of the job above it. Thus, we can construct a feasible chromosome for the 3-level job process scheduling problem with the twin-screwed gene expression.

15	1	5	7	7	12	4	16	3	5	2	3	14	11	8	12	13	6	5	9	10
3	*1*	*3*	*1*	*1*	*3*	*3*	*3*	*2*	*3*	*2*	*2*	*3*	*3*	*2*	*3*	*3*	*3*	*3*	*2*	*2*

Figure 3: The feasible gene expression for a three-level shop scheduling solution.

In order to guarantee the feasibility of a twin-screw-coded chromosome, a specifically designed decoding strategy (shown as Figure 4) is employed to cope with this issue. We employ the example mentioned above to illustrate the process of our proposed decoding method, in which process scheduling information is obtained from lower level to the higher one. The genotype codes of a chromosome are scanned from the beginning to the end; those operations that labeled with a "3" as its implicit twin-code (in the second line) are *Recognized* as part level's jobs and labeled to phenotype; after a round-robin scan, all the jobs belonging to the part level are kicked off from the original chromosome, a second round scan is taken to label the subassembly level, and then the assembly level, until all the levels' operations are labeled and the whole chromosome is decoded to a complete 3-level job scheduling solution with phenotype status.

In the decoding process (Figure 4), ${}^{o5}_{1}$ represents the first operation of job 5; $o4$ represents the operation of job 4. The final decoded solution of the above example is as follows:

(i) part level: $o15$, ${}^{o5}_{1}$, $o12$, $o4$, $o16$, ${}^{o5}_{2}$, $o14$, $o11$, $o12$, $o13$, $o06$, ${}^{o5}_{3}$;

(ii) subassembly level: ${}^{o3}_{1}$, $o2$, ${}^{o3}_{2}$, $o8$, $o9$, $o10$;

(iii) assembly level: $o1$, ${}^{o7}_{1}$, ${}^{o7}_{2}$.

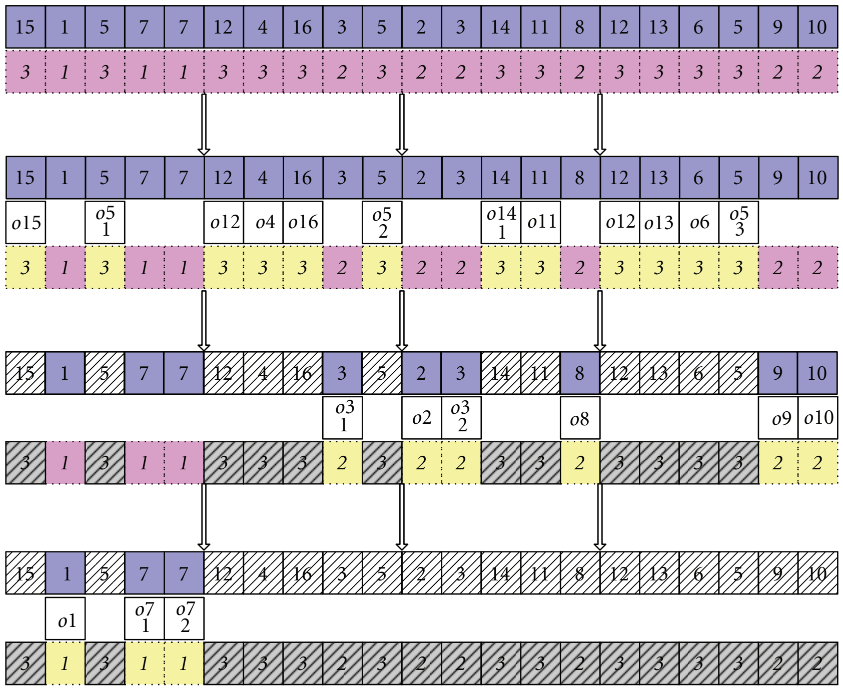


Figure 4: A decoding example.

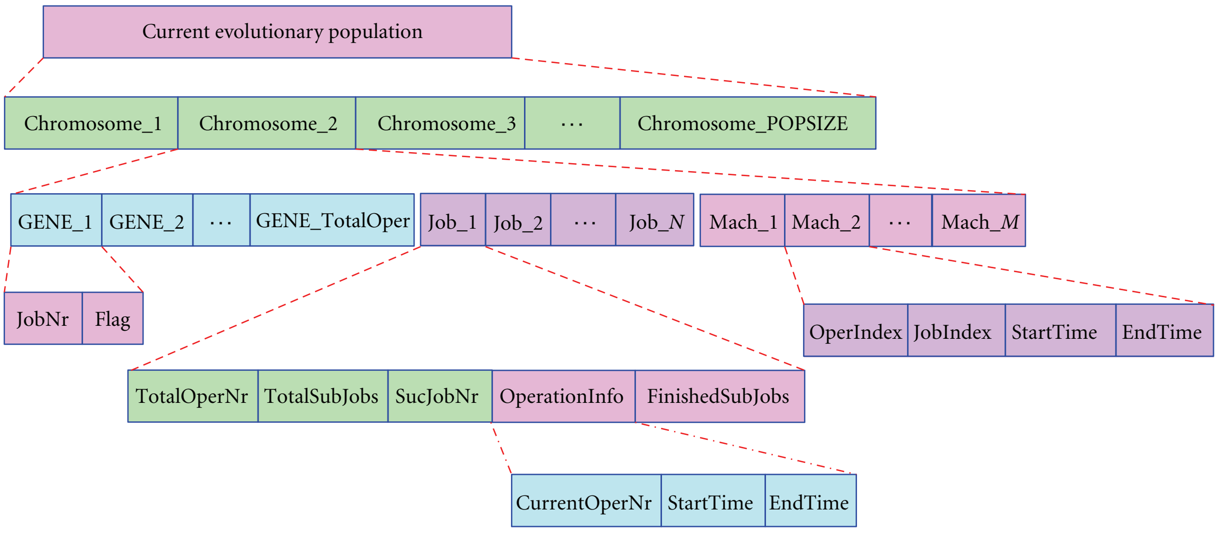


Figure 5: The data structure for decoding process.

In order to promote the decoding efficiency, an object-oriented data structure is designed to fulfill this task (Figure 5). The principle of designing the data structure and some variable abbreviations are noted as the follows.

(1) A chromosome composes three parts: twin-screw codes for sequence and "level" information, job-related information, and machine-related information.

(2) "GENE_i" represents the twin-screw coding structure of a chromosome; "JobNr" represents the job number (the number of each job is equal to its operation number); The initial "Flag" of job "i" indicates the production level, which job "i" belongs to.

During the decoding process, when all the operations of one subjob of job "i" are finished, we add "Job_i"'s "FinishedSubJobs" by (1): when the value is equal to "i"'s "TotalSubJobs", that means all the subjobs of "i" have been finished we turn the "Flag" of "i" from 1 to 0 under this circumstance, and return the scan pointer into the beginning of the twin-screw, and rescan the left operations.

(3) "OperationInfo" records the start time and end time of each job.

(4) "TotalOperNr", "TotalSubJobs", and "SubJobNr" are the initial status of the jobs; these information can be obtained from the configuration document.

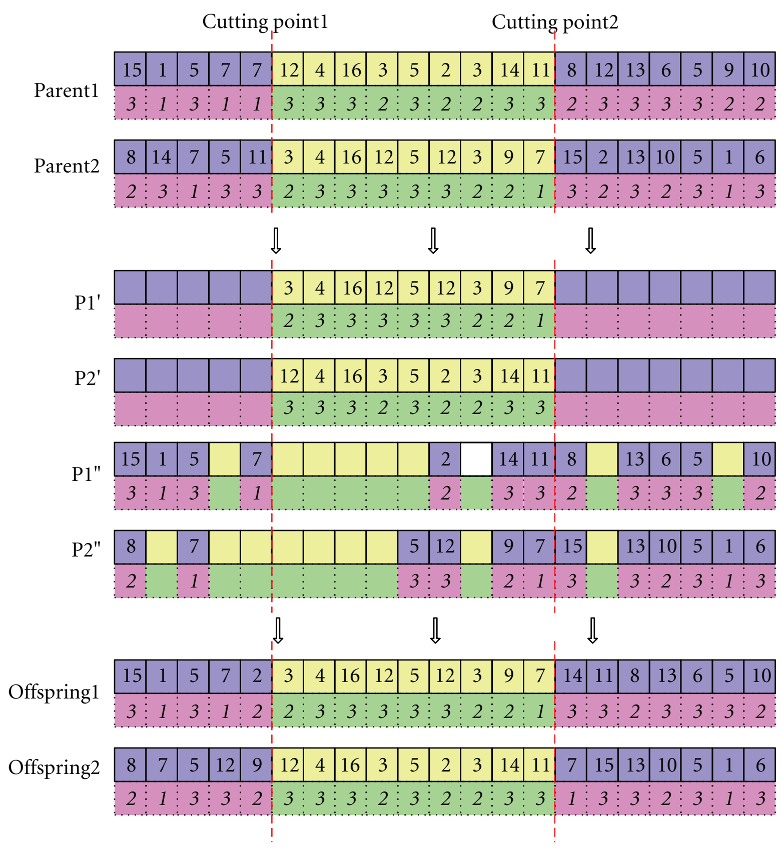


Figure 6: The crossover operator: an example.

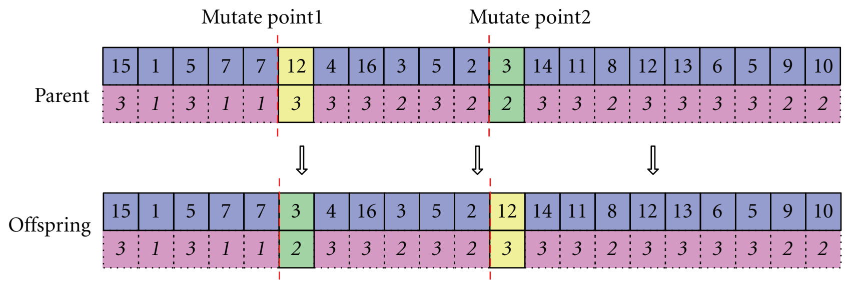


Figure 7: The mutation operator: an example.

(5) "Mach_i" records the job sequences at machine "i"; these pieces of information are employed to verify the validity of a solution, and to make preparation data for Gantt graph, while not participating the decoding process.

4.2. Evolutionary Operators: Crossover and Mutation. In order to guarantee the feasibility and validity of the chromosome during the population evolution, we propose improved crossover and mutation operators for the twin-screw-coded chromosome based on previous work in permutation scheduling problem studies (see [22]). The principle of designing the evolutionary operators is similar to that of PMX crossover and swap mutation operators (see [22]), which can be illustrated by examples shown in Figures 6 and 7.

Crossover operator for a twin-screw-coded chromosome is as follows.

Step 1. Randomly generate two cutting points c_1, c_2 (assume $c_1 < c_2$) on the two parents chromosome $chrom_1$, $chrom_2$.

Step 2. Exchange the partial chromosome between c_1 and c_2 (not only the process sequence information but also the "level" information) to get the two proto-children, shown as P_1' and P_2' in Figure 6.

Step 3. Scan and eliminate the existing elements of P_1' from parent 1 and P_2' from parent 2 to determine the mapping relationship between two mapping sections. After the mapping operation, we get two subsidiary partial mapping information as the P_1'', P_2'' shown in Figure 6.

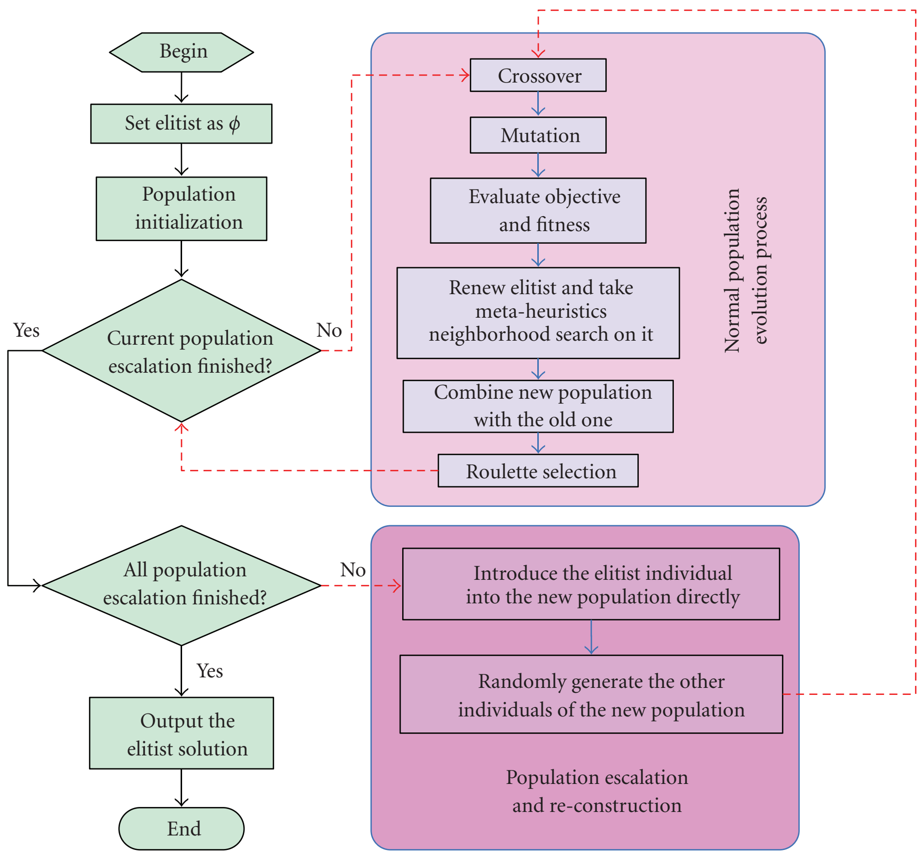

Begin
Set elitist as ϕ
Population initialization
Current population escalation finished?
Yes
No
Crossover
Mutation
Evaluate objective and fitness
Renew elitist and take meta-heuristics neighborhood search on it
Combine new population with the old one
Roulette selection
Normal population evolution process
All population escalation finished?
No
Yes
Introduce the elitist individual into the new population directly
Randomly generate the other individuals of the new population
Population escalation and re-construction
Output the elitist solution
End

FIGURE 8: Flowchart of the Twin-screw Coded Evolutionary Algorithm (TCEA) for MLPS.

Step 4. Legalize the offspring with the mapping relationship information, and obtain two feasible offspring children (*offspring1* and *offspring2* in Figure 6).

Please note that the whole crossover process is guided and dominated by the sequence information, but not the "level" information. Because in our proposed twin-screw-coded chromosome, each bit of "level" information is bound strength with a corresponding operation information, it does not make any sense except in the decoding process.

Mutation operator for twin-screw-coded chromosome is as follows.

Step 1. Randomly generate two mutate points m_1, m_2 on the parent chromosome to be mutated.

Step 2. Swap the two position's sequence and "level" information to generate a legal offspring child (as shown in Figure 7).

4.3. Population Reconstruction with Elitism Strategy. In our previous work, we have proposed an escalating evolutionary structure (shown in Figure 8), which has outperformed several other modern heuristic algorithms with applications to flow shop scheduling problems under the similar computational cost. In order to solve the integrated MLPS model efficiently, we introduce the escalating strategy into the twin-screw-coded EA to enhance its convergence performance. The brief idea of escalating strategy can be explained as follows.

A population evolves from a random beginning status, the probability of an individual to bring its offspring lies on its fitness. After some generations' evolution, the population may keep in evolving with no progress further more in some successive generations; then the elitist individual (the best one in the population from the beginning to current generation) is kept and introduced into a new population directly, and all the other individuals of the new population are generated randomly (reconstruction/reinitialization). Thus, the new population continues the evolution process until the stop criterion is satisfied.

The escalation process implies two meanings: (1), the elitist individual will be introduced into the new population directly; (2), other individuals of the new population will be generated randomly, where (1) makes it possible to utilize the previous level's search information, and (2) is designed to keep the population diversity, which helps the algorithm escaping from premature.

5. Case Study

5.1. Problem Description. In order to evaluate the performance of our proposed modeling and optimization technique, a 3-level production scheduling problem from one of Chinese famous satellite production factory is employed

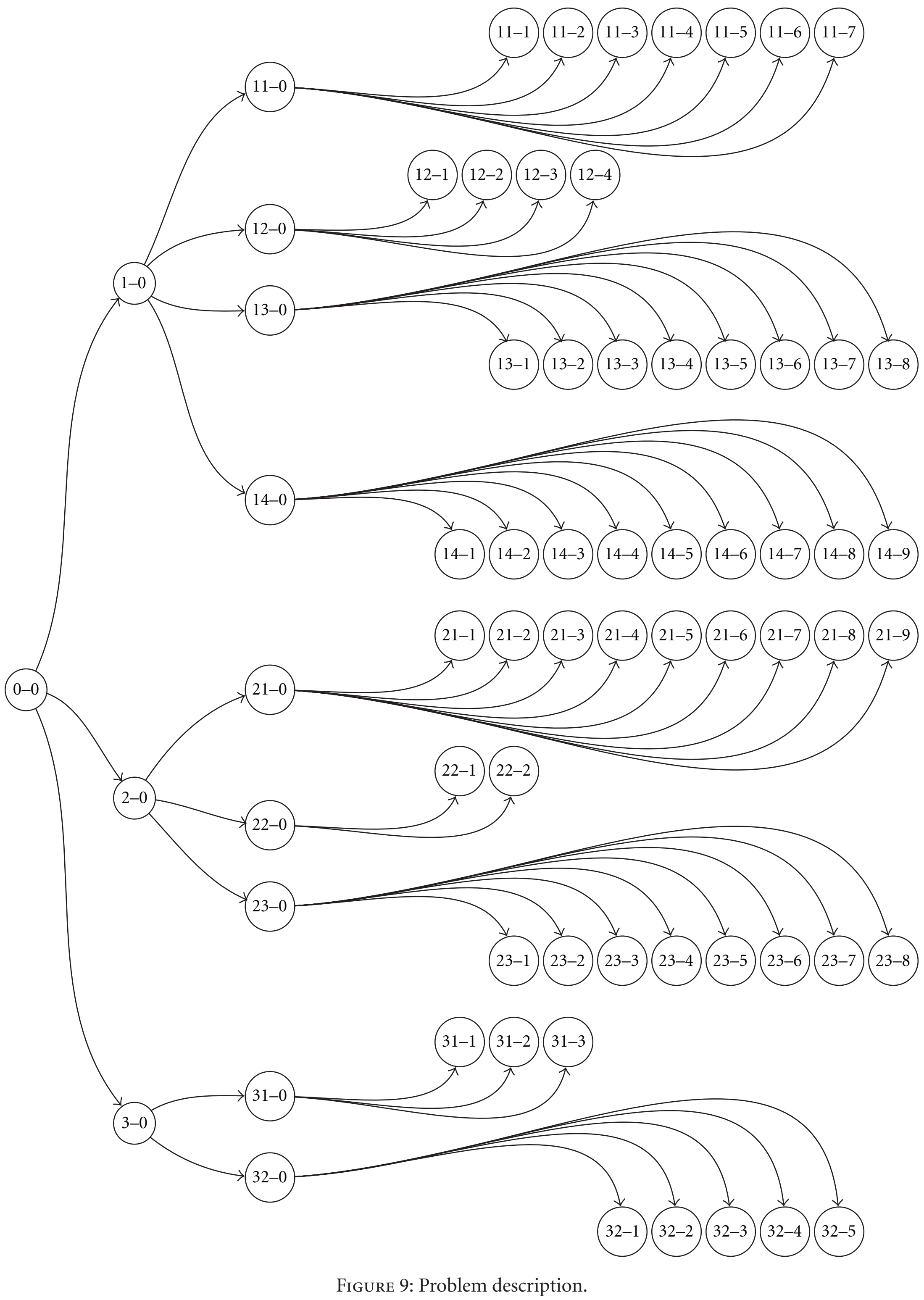


Figure 9: Problem description.

for case study. All the processing information has been necessarily deposed with pre-declassification need beforehand.

The hierarchical product structure of the problem is similar to the model described as (1)–(24): product 0–0 consists of 3 assemblies, each assembly consists of 4, 3, 2 subassemblies, respectively, and each subassembly consists of some given number of parts. The job of a higher level could begin its process operation only if all its sub-products are finished and are assembled. The process information includes technical constraints within levels and between levels, process time, and predefined operational sequence between jobs. The hierarchical logic relationship of the problem can be highlighted as shown in Figure 9.

Table 1: The raw data of case study problem.

	Oper.1		Oper.2		Oper.3		Oper.4		
JobNr	machine	time	machine	time	machine	time	machine	time	· · ·
11–1	O1	16	O4	7.1	O1	16	DM	38	· · ·
11–2	O1	8	RM	5.3	O3	7	PM	10	· · ·
11–3	O2	20	DM	44.8	O2	11.2	PM	80	· · ·
11–4	RM	3.3	O3	0.4	PM	4.6	O1	8	· · ·
11–5	O2	10.4	DM	44.2	O1	16	O2	12	· · ·
11–6	O2	5	DM	14	RM	2	O1	8	· · ·
11–7	O2	20	DM	56	RM	33.2	O3	2	· · ·
11–0	RM	31	O7	20.3	O1	16	PM	12	· · ·
12–1	RM	29	DM	20	O1	8	RM	22	· · ·
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
12–5	O7	7	RM	11	O1	8	O6	12	· · ·
12–0	RM	27	O2	12					
13–1	O1	8	O4	19.5	RM	21.2	O4	2	· · ·
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
13–8	O7	21	RM	33	O1	8	O6	22	· · ·
13–0	RM	36	OM	54	PM	21	O5	8	
14–1	RM	9	O9	60	O5	6			
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
14–9	O2	2.4	O1	8	DM	12.4	O3	2	· · ·
14–0	O2	9	O2	5	O1	8	O2	6	· · ·
1–0	O3	82	O2	14	RM	61	O3	24	· · ·
21–1	O7	27.2	O7	10.4	O1	64	O6	144	· · ·
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
21–9	RM	1152	OM	768	PM	768	O5	384	· · ·
21–0	O3	208	RM	136	DM	84	O5	32	· · ·
22–1	RM	61.5	DM	170	O1	40	DM	720	· · ·
22–2	RM	19	DM	60	O1	16	DM	248	· · ·
22–0	O3	23	RM	54					
23–1	O11	8.5	DM	5.2	O1	8	O9	7	· · ·
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
23–8	O11	8.5	DM	5.2	O1	8	O9	7	· · ·
23–0	O3	43	DM	44	O3	4			
2–0	O3	84	DM	20	O9	32	DM	88	· · ·
31–1	O2	4.6	RM	4	DM	6.4	O3	3	· · ·
31–2	O2	8	O1	8	O2	3.3	DM	16	· · ·
31–3	O2	8	OM	4.4	O5	8	O3	0.5	· · ·
31–0	O2	32	OM	58	O1	8	O2	28	· · ·
32–1	O2	20	OM	12	O3	9.3			
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
32–5	O4	4	DM	1	O3	1			
32–0	O2	5.2	O2	5.2	RM	116	O13	6	· · ·
3–0	O13	40	DM	52	O3	8			
0–0	O3	32	OM	81	O3	23	OM	68	· · ·

Table 1 gives the detail process information of the problem. There are some complementary comments to the problem.

(1) The *O*1–*O*13 in Table 1 represent the process operations on those machines, whose process ability can be greatly increased with a bit costs and the workloads on them can be considered as light as possible. So the capability of these machines can be considered as infinite. These machines include common low-precise manufacturing machines, like lathe, planer, grinder, and so forth.

While the workloads on the other kind of machines are obviously heavy, not only the fixed expensive purchasing costs but also the expensive unit time process costs on them are much higher than the common ones. There are 4 units of such machines in the factory that we investigated, whose name can be listed as Rough Milling machine (RM), Precise Milling machine (PM), Digital Milling machine (DM), and Other Milling machine (OM), respectively.

(2) The definition of job numbers (JobNr) in Table 1 is as follows.

(i) 11–01 represents the 01 part of subassembly 11;

(ii) 11–0 is the label for subassembly 11;

(iii) 1–0 is the label for assembly 1;

(iv) 0–0 is the label of the final product.

After analyzing the process information, we make two assumptions to deduce the computational complexity of the raw problem, in which only the most "expensive" and "crowed" machines are specifically treated, while we neglect the scheduling planing on those "cheap" or "loosely required" machines.

Assumption 1 (Machine Scarce/Nonscarce Assumption). As we know, a satellite product requires more precision than a civil product; its large size and highly precision quality leads to the need of high-performance milling machines in many operations. We discriminately treat the operations that need to be operated on milling machines with those operations that need to be operated on other machines and call the operations on these two kind of machines as *Scarce machine* operations and *nonscarce machine* operations. Consecutive operations on *nonscarce machines* can be combined into "Dummy Operations".

The adjacent operations on *nonscarce machines* can be combined as one operation time, and the new combined operation can be considered as operations that have no constraints on machine availability; we only take its operation time into account but neglect the operation's machine information.

Assumption 2 (Machine Specification Assumption). As different machines for the same type of task in the satellite factory take different costs, the milling operations are allocated to different machines according to its precision requirements, which aims at strengthening the economic profit of the whole. After the hypothesis of dummy operation and machine operation specialization, we focus our effort on the scheduling of scarce machines, which can help us avoid to waste time on insignificant operations or wasting costly machines on simple or nonprecise operations; thus can we possibly obtain a better solution in a given time.

5.2. Data Preprocessing. There are 4 milling machines (RM/PM/DM/OM) in the factory. Since the operations on these milling machines are the bottle neck of the scheduling problem, we allocate the milling operations to these machines with regard to each machine's operational precision and cost.

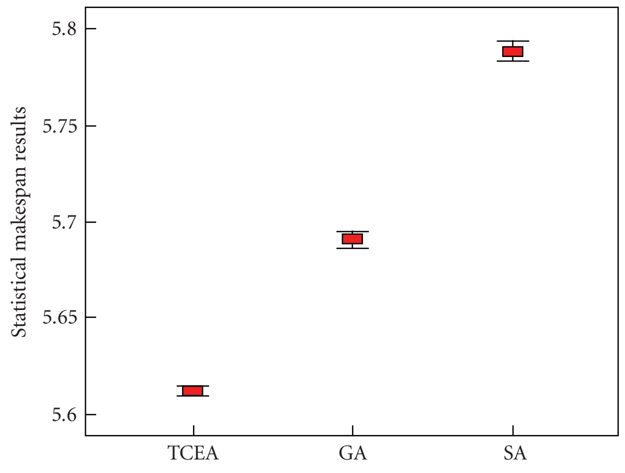


FIGURE 10: Statistical results of 3-level MLPS with TCEA, GA, and SA.

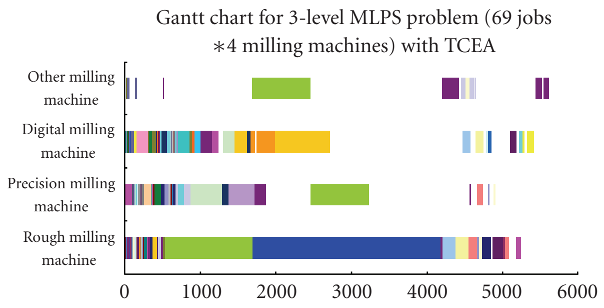


FIGURE 11: A typical solution of 3-level MLPS with TCEA.

(i) Rough milling operations are allocated to the machine "RM".

(ii) Precise milling operations are allocated to the machine "PM".

(iii) Digital milling operations are allocated to the machine "DM".

(iv) Other milling operations are allocated to the machine "OM".

All the other nonmilling operations are considered as "dummy operations" (as mentioned in Section 5.1). After we combined the "dummy operations", we get the new modified process data of the problem (shown in Table 2).

5.3. Comparison Study Algorithms and Parameters Setting. In order to evaluate the performance of our proposed TCEA with its application to MLPS, we employ two basic metaheuristic algorithms, GA (Genetic Algorithm), and SA (Simulated Annealing) as comparison algorithms for case study.

In order to obtain the best performance of TCEA, parameters experiment has been taken to search the best parameters combination. The experiment is designed with the following guidance rules [21].

TABLE 2: The modified data of case study problem.

Level 1: the part level										
JobNr	dum	mach	time	dum	mach	time	dum	mach	time	· · ·
11–01	32.1	DM	38	26	NS	0				
11–02	8	RM	5.3	7	PM	10	8	DM	80	· · ·
11–03	20	DM	44.8	11.2	PM	80	4.8	NS	0	· · ·
11–04	0	RM	3.3	0.4	PM	4.6	8	PM	23	· · ·
11–05	10.4	DM	44.2	28	PM	80	4.4	NS	0	· · ·
11–06	5	DM	14	0	RM	2	11.5	DM	8	· · ·
11–07	20	DM	56	0	RM	33.2	14	DM	32	· · ·
12–01	0	RM	29	0	DM	20	8	RM	22	· · ·
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
12–05	7	RM	11	20	PM	52	1	DM	19	· · ·
13–01	27.5	RM	21.2	2	PM	11	18.4	NS	0	· · ·
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
13–08	21	RM	33	30	PM	52	13	DM	19	· · ·
14–01	0	RM	9	66	NS	0				
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
14–09	10.4	DM	12.4	2	NS	0				
21–01	245.6	PM	416	10.4	DM	152				
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
21–09	0	RM	1152	0	OM	768	0	PM	768	· · ·
22–01	0	RM	61.5	0	DM	170	40	DM	720	· · ·
22–02	0	RM	19	0	DM	60	16	DM	248	· · ·
23–01	8.5	DM	5.2	19.5	DM	13	8	DM	32	· · ·
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
23–08	8.5	DM	5.2	19.5	DM	13	8	DM	32	· · ·
31–01	4.6	RM	4	0	DM	6.4	3	NS	0	· · ·
31–02	19.3	DM	16	2.5	NS	0				
31–03	8	DM	4.4	8.5	NS	0				
32–01	20	OM	12	9.3	NS	0				
· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
32–05	4	DM	1	1	NS	0				
Level 2: the subassembly level										
JobNr	dum	mach	time	dum	mach	time	dum	mach	time	· · ·
11–0	0	RM	31	36.3	PM	12	16	NS	0	· · ·
12–0	0	RM	27	12	NS	0				
13–0	0	RM	36	0	OM	54	0	PM	21	· · ·
14–0	29	DM	29							
21–0	208	RM	136	0	DM	84	32	NS	0	· · ·
22–0	23	RM	54							
23–0	43	DM	44	4	NS	0				
31–0	32	OM	58	36	OM	60	8	OM	12	· · ·
32–0	10.4	RM	116	23	NS	0				
Level 3: the assembly level (including the final product)										
JobNr	dum	mach	time	dum	mach	time	dum	mach	time	· · ·
1–0	96	RM	61	24	NS	0				
2–0	84	DM	20	32	DM	88	16	NS	0	· · ·
3–0	40	DM	52	8	NS	0				
0–0	32	OM	81	23	OM	68				

Table 3: Parameters setting for 3-level MLPS study case.

	pop_size	max_gens	P_c	P_m	n_L
TCEA	200	200×5	0.9	0.1	20
GA	200	1000	0.9	0.1	
	T_0	T_{final}	β	n_L	
SA	1000	0.1	0.999	50	

Table 4: Statistical optimization results of the 3-level MLPS with TCEA, GA and SA.

Algorithms	avg.Makespan	max.Makespan	min.Makespan	dev.Makespan
TCEA	5612.4	5632	5598	3.86
GA	5691.2	5704	5668	8.61
SA	5788.5	5814	5768	11.26

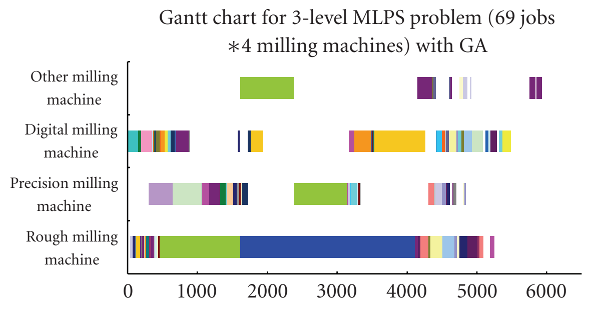


Figure 12: A typical solution of 3-level MLPS with GA.

(i) Population size (POP_SIZE) varies from 100 to 500, skip rule 100.

(ii) Evolutionary generation varies from 100 to 500, skip rule 100.

(iii) Crossover probability varies from 0.3 to 0.9, skip rule 0.1.

(iv) Mutation probability varies from 0.01 to 0.1, skip rule 0.01.

(v) Population escalation gap varies from 10 to 1, skip rule 1.

(vi) Elitist local search step varies from 10 to 50, skip rule 10.

After the parameters experiment, we set the parameters of TCEA as in Table 3. In order to compare the performance of TCEA with that of GA and SA in a fair circumstance, we make the similar parameters experiments for GA and SA, respectively, in which the total CPU time consumption is kept in the same level as TCEAs. After the experiments, we can set the parameters of TCEA, GA, and SA as in Table 3. Since the twin-screw coding strategy is a general encoding strategy designed for MLPS problems, we employ the coding strategy in all the three algorithms.

5.4. Results Analysis. Since all the algorithms that we study are metaheuristic algorithms, we run each algorithm for 20 independent times to collect their statistical results. With parameters set in Table 3, we get the optimization results as in Table 4 and figure 10, in which the average result of TCEA outperforms that of GA and SA:

The average makespan of product 0–0 obtained by our TCEA is 5612.4, and the results of GA and SA are 5691.2 and 5788.5, respectively; all these metaheuristic algorithms outperformed current technique in the factory (6580).

However, the computational cost of TCEA (about 270s) is a bit longer than that of GA (about 250s) and SA (about 190s) in the same experiment environment (all the experiments are taken in a CPU Pentium IV-3.2 G, 1 G Ram PC platform).

In general, the statistical results show the outstanding performance of our TCEA by contrast to that of GA and SA to cope with an MLPS problem.

Figures 11 and 12 show two typical solutions derived from TCEA and GA, respectively.

6. Conclusion

A twin-screw-coded evolutionary algorithm (TCEA), which is motivated by solving a typical multi-level production scheduling problem (MLPS), is put forward in this paper. The principle of the new algorithm is introduced; besides, a real 3-level satellite part's case study has revealed the superiority of TCEA by contrast to GA and SA, which further demonstrates the effectiveness and practicability of our integrated model and optimization technique in solving such complex production scheduling problems. As we know, MLPS is a complex NP-hard problem; this work just shows the preliminary result of our project. Further research has been taken, in which multiobjective MLPS problem modeling and optimization technique has been taken into consideration.

Acknowledgments

The authors would like to thank the Natural Science Foundation of China (Grant nos. 70771003, 60974051), the Natural Science Foundation of Hebei Province (Grant no. F2010001714), the Doctorial Faculty Research Foundation of North China Electric Power University (Grant no. 200822025), and Shanghai Special Foundation of

Outstanding Young Teachers in University (Grant no. ssc08023) for their financial supports with this research. The authors would like to thank Dr. X.W. Tan and J. Wang for their helpful suggestions to improve this work, and thank Ms. S.P. Wang for her help in case study data collection. The authors would also like to thank the anonymous reviewers and Dr. C.K. Ting for their constructive comments to improve the work.

References

[1] D. Biskup, "A state-of-the-art review on scheduling with learning effects," *European Journal of Operational Research*, vol. 188, no. 2, pp. 315–329, 2008.

[2] N. Boysen, M. Fliedner, and A. Scholl, "Sequencing mixed-model assembly lines: survey, classification and model critique," *European Journal of Operational Research*, vol. 192, no. 2, pp. 349–373, 2009.

[3] Y. Li, K. F. Man, K. S. Tang, S. Kwong, and W. H. Ip, "Genetic algorithm to production planning and scheduling problems for manufacturing systems," *Production Planning and Control*, vol. 11, no. 5, pp. 443–458, 2000.

[4] W. Shen, Q. Hao, H. J. Yoon, and D. H. Norrie, "Applications of agent-based systems in intelligent manufacturing: an updated review," *Advanced Engineering Informatics*, vol. 20, no. 4, pp. 415–431, 2006.

[5] Y. K. Kim, K. Park, and J. Ko, "A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling," *Computers and Operations Research*, vol. 30, no. 8, pp. 1151–1171, 2003.

[6] L. Li, J. Y. H. Fuh, Y. F. Zhang, and A. Y. C. Nee, "Application of genetic algorithm to computer-aided process planning in distributed manufacturing environments," *Robotics and Computer-Integrated Manufacturing*, vol. 21, no. 6, pp. 568–578, 2005.

[7] F. Riane, A. Artiba, and S. Iassinovski, "An integrated production planning and scheduling system for hybrid flowshop organizations," *International Journal of Production Economics*, vol. 74, no. 1–3, pp. 33–48, 2001.

[8] L. Tang, J. Liu, A. Rong, and Z. Yang, "A review of planning and scheduling systems and methods for integrated steel production," *European Journal of Operational Research*, vol. 133, no. 1, pp. 1–20, 2001.

[9] N. Vandaele and L. De Boeck, "Advanced resource planning," *Robotics and Computer-Integrated Manufacturing*, vol. 19, no. 1-2, pp. 211–218, 2003.

[10] W. Xia and Z. Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems," *Computers and Industrial Engineering*, vol. 48, no. 2, pp. 409–425, 2005.

[11] J. Deschamps and J. Bourrieres, "Multi-level data model for load allocation to distributed manufacturing resources," in *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 357–362, New York, NY, USA.

[12] M. Pinedo, *Scheduling-Theory, Algorithms, and Systems*, Prentice Hall, Upper Saddle River, NJ, USA, 1995.

[13] D. E. Shobrys and D. C. White, "Planning, scheduling and control systems: why cannot they work together," *Computers and Chemical Engineering*, vol. 26, no. 2, pp. 149–160, 2002.

[14] C. Moon and M. Gen, "Genetic algorithm-based approach for design of independent manufacturing cells," *International Journal of Production Economics*, vol. 60, pp. 421–426, 1999.

[15] A. Drexl and A. Kimms, "Lot sizing and scheduling—survey and extensions," *European Journal of Operational Research*, vol. 99, no. 2, pp. 221–235, 1997.

[16] B. Karimi, S. M. T. Fatemi Ghomi, and J. M. Wilson, "The capacitated lot sizing problem: a review of models and algorithms," *Omega*, vol. 31, no. 5, pp. 365–378, 2003.

[17] M.-J. Yao and J.-X. Huang, "Solving the economic lot scheduling problem with deteriorating items using genetic algorithms," *Journal of Food Engineering*, vol. 70, no. 3, pp. 309–322, 2005.

[18] E. Alvarez, "Multi-plant production scheduling in SMEs," *Robotics and Computer-Integrated Manufacturing*, vol. 23, no. 6, pp. 608–613, 2007.

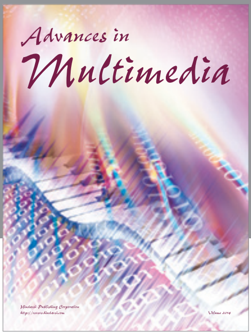
[19] B. R. Sarker and C. V. Balan, "Operations planning for a multi-stage kanban system," *European Journal of Operational Research*, vol. 112, no. 2, pp. 284–303, 1999.

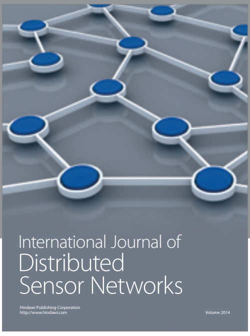
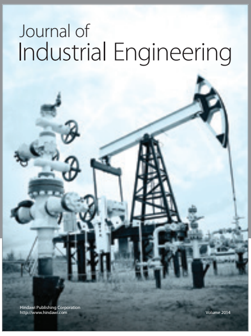
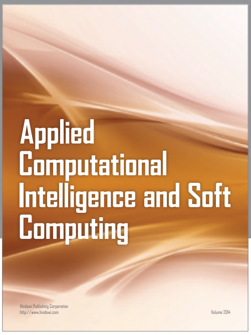
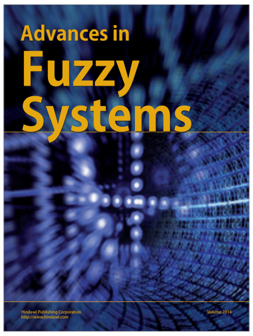
[20] Y.-N. Yang, H. R. Parsaei, and H. R. Leep, "Prototype of a feature-based multiple-alternative process planning system with scheduling verification," *Computers and Industrial Engineering*, vol. 39, no. 1-2, pp. 109–124, 2001.

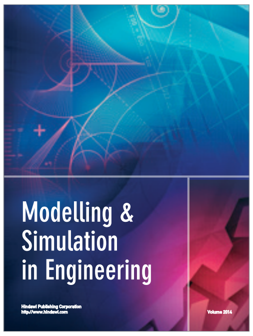
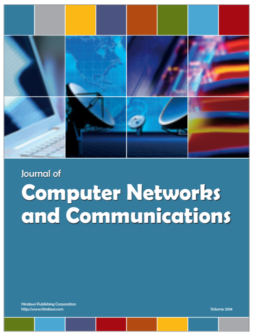
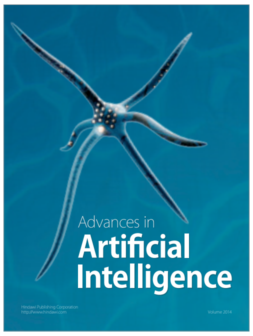
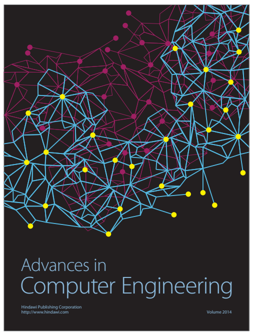
[21] T. P. Bagchi, *Multiobjective Scheduling by Genetic Algorithms*, Kluwer Academic, Boston, Mass, USA, 1999.

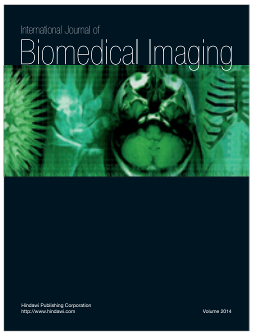
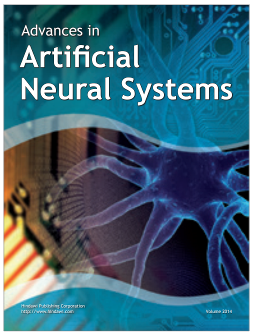
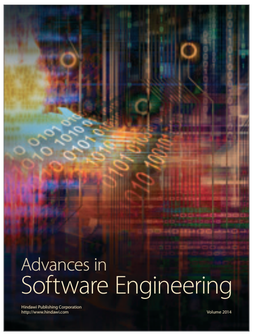
[22] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*, John Wiley & Sons, New York, NY, USA, 1996.

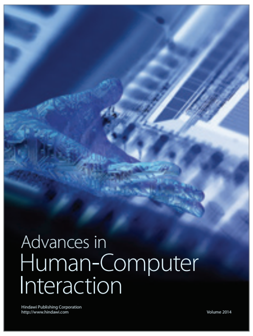
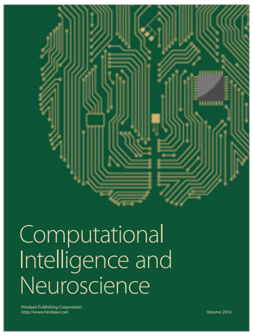
[23] T. Murata, *Genetic algorithms for multi-objective optimization*, Ph.D. thesis, Osaka Prefecture University, Osaka, Japan, 1997.


Advances in
Multimedia

The Scientific World Journal

International Journal of
Distributed Sensor Networks

Journal of
Industrial Engineering

Applied Computational Intelligence and Soft Computing

Advances in
Fuzzy Systems


Modelling & Simulation in Engineering

Journal of
Computer Networks and Communications

Advances in
Artificial Intelligence

Advances in
Computer Engineering

International Journal of
Computer Games Technology

International Journal of
Biomedical Imaging

Advances in
Artificial Neural Systems

Advances in
Software Engineering

Journal of
Robotics

Advances in
Human-Computer Interaction

Computational Intelligence and Neuroscience

International Journal of
Reconfigurable Computing

Journal of
Electrical and Computer Engineering