

Article

# Integrated Multi-Project Scheduling and Hierarchical Workforce Allocation in the ETO Assembly Process

Chun Jiang <sup>1</sup> , Xiaofeng Hu <sup>1,\*</sup> and Juntong Xi <sup>1</sup>

<sup>1</sup> School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China; sjjzxc@163.com (C.J.); jtxi@sjtu.edu.cn (J.X.)

\* Correspondence: wshxf@sjtu.edu.cn; Tel.: +86-021-3420-5694

Received: 14 January 2019; Accepted: 25 February 2019; Published: 1 March 2019



**Featured Application:** With the increasing demand for customization, the engineer-to-order (ETO) production strategy plays an increasingly important role in today's manufacturing industry. The most important phase of ETO production is assembly. Project scheduling and workforce allocation have an impact on the efficiency and cost of the assembly of ETO products and the two problems are interrelated. To the best of our knowledge, this is the only work about project scheduling and workforce allocation in the ETO assembly process. The method proposed in this work can improve the efficiency of the ETO assembly system and has been successfully applied at the Shanghai Electric Power Plant Equipment Company, which is a joint venture between Shanghai Electric Company and Siemens.

**Abstract:** The engineer-to-order (ETO) production strategy plays an important role in today's manufacturing industry. This paper studies integrated multi-project scheduling and hierarchical workforce allocation in the assembly process of ETO products. The multi-project scheduling problem involves the scheduling of tasks of different projects under many constraints, and the workforce allocation problem involves assigning hierarchical workers to each task. These two problems are interrelated. The task duration depends on the number of hierarchical workers assigned to the task. We developed a mathematical model to represent the problem. In order to solve this issue with the minimization of the makespan as the objective, we propose a hybrid algorithm combining particle swarm optimization (PSO) and Tabu search (TS). The improved PSO is designed as the global search process and the Tabu search is introduced to improve the local searching ability. The proposed algorithm is tested on different scales of benchmark instances and a case that uses industrial data from a collaborating steam turbine company. The results show that the solution quality of the hybrid algorithm outperforms the other three algorithms proposed in the literature and the experienced project manager.

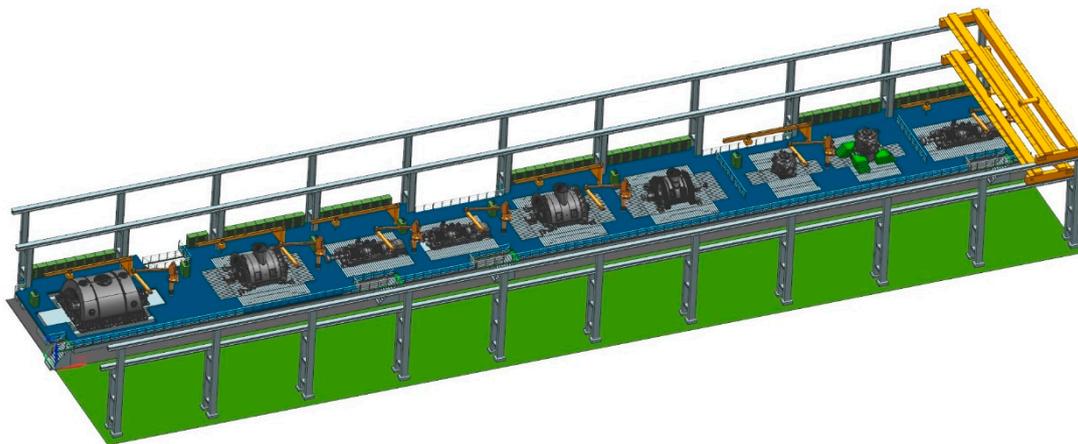
**Keywords:** engineer-to-order; multi-project scheduling; hierarchical workforce; worker allocation; particle swarm optimization; Tabu search

## 1. Introduction

In today's competitive manufacturing industry, there is a constantly increasing demand for customized products, especially within advanced, capital intensive, large-equipment industries [1]. In order to respond to this demand, companies must manufacture and assemble based on specific customer requirements. This type of manufacturing strategy is called engineer-to-order (ETO). ETO products are manufactured and assembled in low volumes to satisfy individual customer's specifications [2]. Typical products include steam turbines and boilers for the power generation industry [3]. A typical ETO product can take months to years to complete and it has strict, complex

ordered assembly relationships between components [4]. The customers of ETO products are very strict with delivery time, and late delivery can result in huge penalties. ETO companies need to ensure that products are delivered in a timely manner [5].

In ETO companies, the most important phase is assembly [6]. Assembly accounts for almost 50% of the total production time, for 20% of the total production cost, and for 30% to 50% of the labor cost [7]. Due to the complexity of assembly, the assembly process of each product is treated as a unique engineering project with necessary lead-time [8]. That is the reason why we use the project scheduling methodology in this study [9]. One company performs not just one project, but two or more projects in parallel. Some projects may have to be simultaneously executed in a certain time period. The projects are independent of each other, but they are located at different work centers, competing for workers with different qualifications, working space, equipment, and tools. The layout of a typical ETO product assembly shop is shown in Figure 1. For a large company, multiple projects planned on the same timeline may have thousands of tasks, and frequent conflicts are inevitable. The high level of customization and long task duration of ETO products requires the production plan to be defined and the details disclosed. It is necessary to schedule and balance the utilization of resources under the guarantee of meeting the delivery deadline of all products [4].



**Figure 1.** The layout of a typical engineer-to-order (ETO) product assembly shop.

In order to react to the high complexity and the large amount of customization in ETO products, the assembly of ETO products is mainly performed by a group of workers. The working situation of the typical ETO product assembly shop is shown in Figure 2. The workers have a lot of experience and are familiar with the assembly process. The task duration is not defined in advance, but depends on the number and the skill level of the workers assigned to the task. There is a downward substitutability in the hierarchical nature of the workforce such that a higher qualified worker type can substitute for a lower qualified one with a higher cost, but not vice-versa [10]. As tasks in the ETO assembly process are manual ones assisted by hand tools or light-weight equipment, the workforce resource has replaced capital, facilities, raw material, and other kinds of manufacturing resources to become the most important resource in the ETO assembly process. If workforce resources can be utilized efficiently, the idle time and production cost can be reduced. The importance of the workforce has been widely recognized, especially in labor-intensive industries. Usually, companies that manufacture ETO products have some kind of estimates of the work content of the assembly tasks based on historical data or the analysis of the work content, and the work content estimate is chosen as the basis in our study [11].



**Figure 2.** Working situation of the typical engineer-to-order (ETO) product assembly shop.

Project scheduling is well-known as the project sequencing problem and one of the most important areas in the manufacturing process [12]. The scheduling of ETO products is much more difficult than that of mass production, because different ETO products have different processing routes, different structures, and different due dates, and multiple products need to be produced in parallel. The literature for project scheduling in ETO companies is scarce [13]. To the best of our knowledge, there is only one earlier work; Alfieri et al., studied single-project scheduling in an ETO environment [8]. The project scheduling problem and the hierarchical workforce allocation problem are interrelated as the task durations depend on the hierarchical workers assigned to each task. In order to solve these problems, decisions about both multi-project scheduling and hierarchical workforce allocation need to be dealt with together. In real production, the integrated problem is considered by the experienced project manager. The project manager makes the project plan and allocates the workers based on experience. The plan is suboptimal because the project manager can't take global information into account.

There is some previous work focusing on the integrated project scheduling and worker allocation issues. Hytonen et al. used discrete event simulation to optimize workforce allocation in assembly lines for highly customized and low-volume products [11]. Heimerl and Kolisch presented the mixed-integer linear program with a tight Linear Programming bound (LP-bound) to solve the problem of simultaneously scheduling IT projects and assigning multi-skilled human resources [14]. Mencía et al. used the genetic algorithms for single-project scheduling with multi-skilled operators [15]. Karam et al. presented the mixed integer linear program to solve the integrated single-project scheduling and multi-skilled workforce allocation issue [16]. Oztemel and Selam used the bees algorithm for multi-mode, resource constrained project scheduling in the molding industry [9]. Lian et al. presented the non-dominated sorting genetic algorithm II (NSGA-II) to solve hierarchical worker allocation and task assignment in a seru production system [17]. Most of the literature treats the problem as a special class of the project scheduling problem, called the multi-mode resource-constrained project scheduling problem (MRCPSP), which is a complex non-deterministic polynomial-time hard (NP-hard) problem [18]. The previous works concentrate on single-project scheduling and hierarchical workforce allocation which is not suitable for ETO assembly. As the ETO projects are assembled in parallel,

we extend the problem as the multi-mode resource-constrained multi-project scheduling problem (MRCMPSP). The MRCMPSP has considerable practical relevance, especially in the manufacturing industry. It is a large combinatorial optimization problem, but it had not received much attention until Wauters et al., [19] choose the subject of the Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA) Challenge 2013. A large international audience was reached: 21 teams from 14 countries and three continents participated in the competition [20–22].

The characteristics of the specified problem are defined as follows: the predecessor and successor relationships of each task in the projects are known, the due date and release time of each project are known, the total number of workers is constrained, and the task duration of each execution mode (worker team) is known. We also consider some features in the ETO assembly process to which the necessary attention has not been devoted in the research up until now, such as that floor space limits the number of concurrently-executed tasks, since the maximum number of workers that can be assigned to each task is limited by work-space. The decision problem is to determine the sequence of tasks and allocate the workers to each task in a multi-project environment in order to minimize the maximum completion time, i.e., the makespan. The scheduling problems are computationally complex and it is difficult to find an optimal solution in a reasonable amount of time. The exact methods can only be used to solve small projects which usually have less than 60 tasks [23].

Metaheuristics are efficient computational methods designed to solve hard combinatorial optimization problems. The previous results show that metaheuristics are able to outperform all good heuristics and usually give high-quality solutions in a reasonable computational time, even for large-size problems. These metaheuristic algorithms can be divided into two categories, population-based algorithms and local search algorithms. The mostly widely used population-based algorithms include particle swarm optimization (PSO), the genetic algorithm (GA), and so on. The most popular local search algorithms include Tabu search (TS), simulated annealing (SA), and so on. Generally speaking, a population-based algorithm shows more global search ability, whereas a local search algorithm has more local search ability. The methods mentioned above have their different metrics and shortcomings. If we combine several methods properly, a better balance between quality and efficiency of solutions may be achieved [24–26]. Considering the convergence ability of PSO and the exploitation ability of TS, we propose a hybrid discrete algorithm with combined PSO and TS to solve this problem.

To the best of our knowledge, the integrated multi-project scheduling and hierarchical workforce allocation in the ETO assembly process has not been tackled before in the existing literature. This has been a motivation of the current work. The rest of this paper will be organized as follows: Section 2 provides the mathematical model for the problem, and Section 3 describes the hybrid algorithm. In Section 4, the hybrid algorithms and three other metaheuristic algorithms proposed in the literature are applied to solve benchmark instances selected from the literature and test results are analyzed. Section 5 represents the industrial application. Concluding remarks and future study directions are given in Section 6.

## 2. Mathematical Formulation

The decision problem concerns the scheduling of tasks and the allocation of workers to the proper task in a multi-project environment in order to minimize the maximum completion time, i.e., the makespan.

In this study, we consider the problem subject to the following assumptions:

(1) Manual assembly processes are assumed to be carried out by a worker team. A mode represents a task-worker team with a constant duration.

(2) During the execution of each task, the assigned mode cannot be changed, i.e., preemption is not allowed during the execution of each task.

(3) The precedence relationships of each project forces each task to be scheduled after all precedence tasks.

- (4) The projects are independent of each other.
- (5) Each worker cannot be allocated to more than one task at the same time.
- (6) In the worker team of each task, workers start meanwhile and finish meanwhile until the task is completed.
- (7) The maximum number of workers in each task is constrained by the work-space.
- (8) The set-up time for each task is included in the task duration, and the transportation times of workers between the tasks are negligible.
- (9) Breakdowns are not considered.

The notation used in this section can be summarized as follows:

**Indices**

$I$	Set of projects
$J_i$	Set of tasks for project $i \in I$
$Q$	Set of maximum number of tasks for each project that can be executed concurrently due to floor space constraints
$M_{ij}$	Set of task execution modes in task $j \in J_i$ , which correspond to the worker team
$K$	Set of hierarchical levels
$T$	Set of time periods
$W$	Set of workers

**Parameters**

$r_i$	release date of project $i \in I$ , i.e., the earliest time that tasks of project $i \in I$ can start
$d_i$	due date of project $i \in I$
$f_i$	end time of project $i \in I$
$pred(j)$	the predecessor set of tasks $j$ , i.e., $pred(j) = \{j'   j' \prec j\}$
$w_{ijmax}$	maximum number of workers in task $j \in J_i$
$w_{ijmin}$	minimum number of workers in task $j \in J_i$
$w_k$	number of type- $k$ workers
$z_{mk}$	number of type- $k$ workers in mode $m \in M_{ij}$
$d_{ijm}$	processing time of task $j \in J_i$ in mode $m \in M_{ij}$
$dur_{ij}$	processing time of task $j \in J_i$
$f_{ij}$	end time of task $j \in J_i$

**Decision Variables**

$S_{ij}$	start time of task $j \in J_i$
----------	--------------------------------

$$x_{ijt} = \begin{cases} 1, & \text{if task } j \text{ is performed at time } t \in (0, T] \\ 0, & \text{otherwise} \end{cases}$$

$$y_{ijm} = \begin{cases} 1, & \text{if task } j \text{ is executed in mode } m \in M_{ij} \\ 0, & \text{otherwise} \end{cases}$$

In the multi-project environment, the delay of any project will lead to iterations and alterations of related follow-up work, so we assume that the objective is to minimize the completion time of all projects but not a certain project. We should consider the precedence constraints, release date constraint, due date constraint, work-space constraint, and floor space constraint. Under the assumptions and notations, the mathematical model for the problem is defined as follows:

**Objective Function:**

$$\min \left[ \max_{i \in I} (f_i) - \min_{i \in I} (r_i) \right] \tag{1}$$

**Constraint Conditions:**

$$S_{ij} \geq r_i \quad \forall i \in I, j \in J_i \tag{2}$$

$$S_{ij} < d_i \quad \forall i \in I, j \in J_i \tag{3}$$

$$S_{ij} + \sum_{m=1}^{M_{ij}} y_{ijm} \cdot d_{ijm} \leq D_i \quad \forall i \in I, j \in J_i \tag{4}$$

$$w_{ijmin} \leq \sum_{k=1}^K y_{ijm} \cdot z_{mk} \leq w_{ijmax} \quad \forall i \in I, j \in J_i, m \in M_{ij} \tag{5}$$

$$\sum_{i=1}^I \sum_{j=1}^J \sum_{m=1}^{M_{ij}} x_{ijt} \cdot y_{ijm} \cdot z_{mk} \leq w_k \quad \forall t \in 0, T, \forall k \in K \tag{6}$$

$$S_{ij'} + \sum_{m=1}^{M_{ij'}} y_{ij'm} \cdot d_{ij'm} \leq S_{ij} \quad \forall i \in I, j \in J_i, j' \in pred(j) \tag{7}$$

$$\sum_{j=1}^J x_{ijt} \leq q_i \quad \forall i \in I, \forall t \in 0, T \tag{8}$$

$$\sum_{m=1}^{M_{ij}} y_{ijm} = 1 \quad \forall i \in I, j \in J_i \tag{9}$$

$$dur_{ij} = \sum_{m=1}^{M_{ij}} y_{ijm} \cdot d_{ijm} \quad \forall i \in I, j \in J_i, m \in M_{ij} \tag{10}$$

$$f_{ij} = s_{ij} + dur_{ij} \quad \forall i \in I, j \in J_i \tag{11}$$

where the objective function is the minimization of the total makespan, the duration of the whole multi-project schedule. Constraint (2) ensures that the start time of the task for each project should be greater than or equal to the release date of project. Constraint (3) ensures that the start time of the task for each project should be less than the due date of project. Constraint (4) ensures that each project needs to be completed before the due date. Constraint (5) implies that the number of workers assigned to each task must be within certain limits due to the work-space constraint. Constraint (6) implies that at any time, the number of type-*k* workers assigned to the tasks should be lower than or equal to the total number of type-*k* workers. Constraint (7) ensures that the precedence relationships between tasks of the same project cannot be violated, and the starting time of the task should be greater than or equal to the finishing time of its preceding set of tasks. Constraint (8) implies that the number of tasks that can be executed concurrently for each project should be limited due to floor space constraints. Constraint (9) ensures that each task is being performed only in one mode. Constraint (10) denotes that the processing time of a task is equal to the processing time of the assigned execution mode for that task. Constraint (11) implies that the end time of the task is equal to the start time plus the processing time, which means the task is non-preemptive.

### 3. Solution Procedure

Particle swarm optimization (PSO) is a population-based stochastic optimization algorithm introduced by Eberhart and Kennedy [27,28]. Because of its easy implementation, robustness, and fast convergence ability, PSO has been applied in solving many combinatorial problems. In the PSO system, each individual, called a particle, represents a point and the population, called a swarm, represents a set of points in the search space. A swarm of initial particles are generated and the particles fly through the search space using a velocity function. The *i*-th particle in *d*-dimension solution space is denoted by  $X_i = (x_{i1}, x_{i2}, \dots, x_{id})$ . The *i*-th particle is assigned a randomized velocity  $V_i = (v_{i1}, v_{i2}, \dots, v_{id})$  and is iteratively moved through the problem space. During the evolution phase, each particle follows the two best values: the local best solution, which is the best solution that the *i*-th particle has achieved so far, and the global best solution, which is the best solution obtained by the population so far. Therefore, the velocity function is continually updated in accordance with the particle's previous experience and the group's experience. In spite of the simple concept, reasonable computational time, and few

parameters to adjust, the shortcoming of PSO is that it is easily trapped into local optima and premature convergence. Since the positions and velocities of standard PSO are real-valued, it is designed for solving continuous functions. However, our problem is set in a discrete space and the standard PSO is not suitable, and we should modify the position representation of PSO from continuous to discrete. Thus, we introduce the crossover operator in the classical GA into the PSO to help the individuals to share information among particles. Tabu search (TS), proposed by Glover in 1986, is a famous local search algorithm used to deal with combinational optimization problems [29]. TS uses the Tabu list to prevent being trapped into local optima and the aspiration rule to exploit a prohibited resolution.

In this study, we propose a hybrid discrete algorithm combining the merits of PSO and the TS algorithm to solve this problem. In the proposed algorithm, PSO is used for the global searching process and TS is employed for the local searching process. The proposed algorithm consists of two phases; the first phase of the hybrid algorithm is the PSO algorithm, whose solution is treated as the initial solution for the Tabu search algorithm in the second phase. The framework of the proposed algorithm is shown in Figure 3. The pseudo-code of the hybrid algorithm is shown in Figure 4.

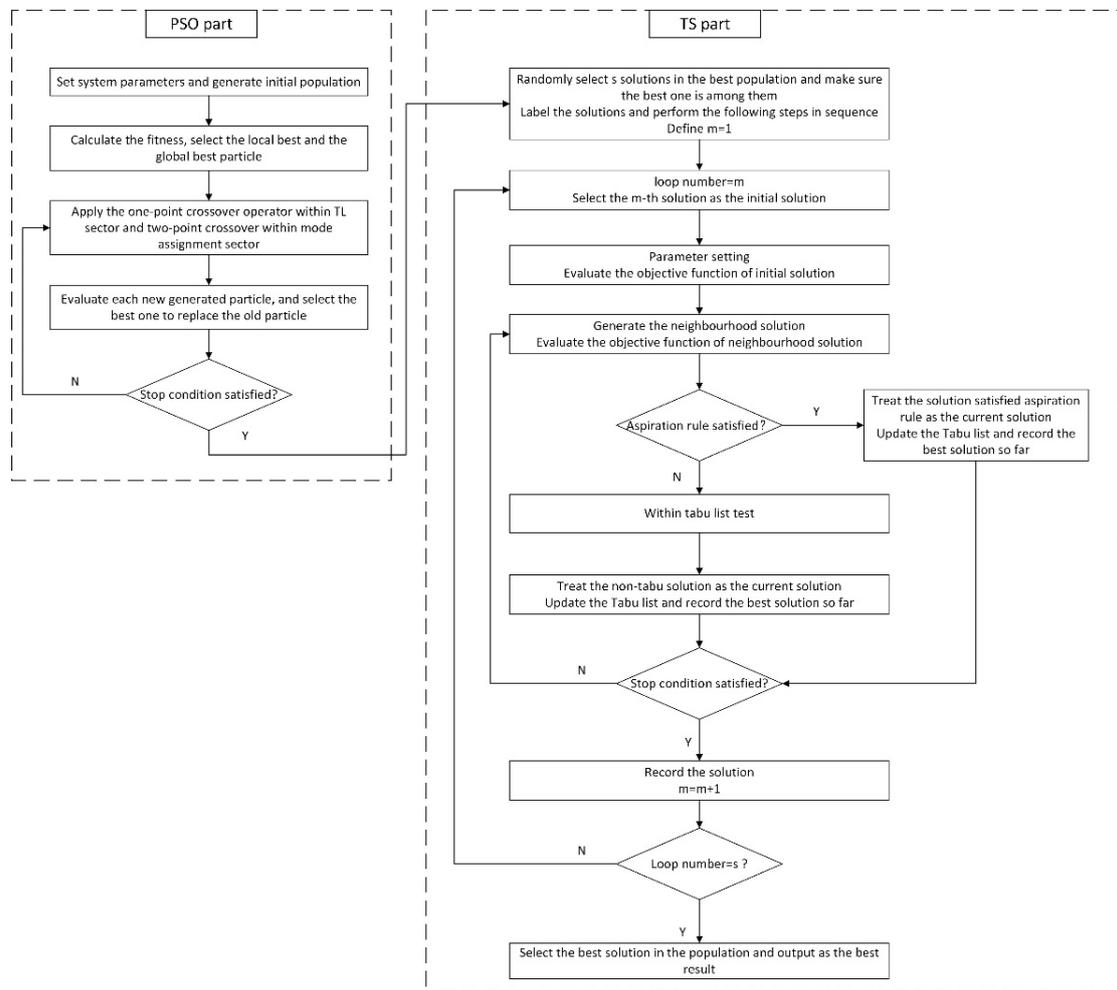


Figure 3. Framework of the hybrid algorithm.

### 3.1. Initialization

#### 3.1.1. Combining All the Projects and Relabeling the Tasks

We combined all the projects into a combined precedence graph and relabeled the tasks. If the first project had  $n$  tasks, then the first task of the second project was relabeled as  $n + 1$ , and so on.

```

Input: precedence relationship, task duration, mode assignment,
       worker resource constraints, space constraints
Output: starting time and mode assignment of each task
Do Preprocessing
Generate initial population
For each particle i
    Set i as the i-local best particle(i-pbest);
    Calculate the fitness value for i;
End
Choose the particle with the best fitness value as the global best particle (gbest)
Do
    For each particle i
        apply crossover operator between i and i-pbest to generate i-temp;
        if i-temp is better than i-pbest in fitness value
            replace i-pbest with i-temp;
        end if
        apply crossover operator between i-temp and gbest to generate i-final;
    End for
    Select the particle with best fitness value in finals as temp final
    if temp final is better than gbest in fitness value
        replace gbest with temp final;
    end if
While maximum iterations is not attained
Select s solutions in the pbest population and make sure the gbest is among them
Label the solutions
For m=1 to s
    treat the m-th solution as initial solution
    calculate the fitness value;
    while not stop
        Generate the neighbors of solution i;
        calculate the fitness value;
        update the Tabu list and record the best solution so far
    end while
Output the best solution
End

```

**Figure 4.** The pseudo-code of the hybrid algorithm.

### 3.1.2. Encoding and Decoding

We adopted the chromosome encoding method. The length of chromosome was twice the number of total tasks. The chromosomal representation was comprised of two vectors; the first vector was the precedence feasible task list (TL) for the scheduling process, while the second vector was the mode assignment for task execution. The TL was a precedence feasible permutation of tasks, in which each task must occur after all its predecessors and before all its successors. This structure was called the task list. The second vector was a list of execution modes for all tasks, and the *k*-th element of this list defined the execution mode of task *k*. Each chromosomal representation determined the sequence of tasks for each project and the mode for each task. We show an example in Figure 5; task 5 is to be executed firstly and its corresponding mode is 3, the mode of task 7 is 1 and so on. The advantage of this encoding scheme is that it can decode easily and improve the execution speed. We applied the serial schedule generation scheme (serial SGS) to generate the schedule related to individuals, which consisted of precedence feasible TL with the mode assignment.

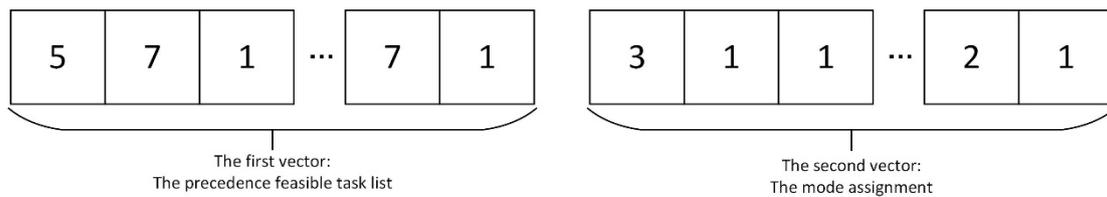


Figure 5. The example chromosome.

### 3.2. PSO Phase

#### 3.2.1. Initial Population

In this approach, each particle in the population was generated randomly without any experience. This approach had merits and shortcoming. The merits were that it was simple to implement and to keep the diversity of population, the shortcoming was that it had less experience.

#### 3.2.2. Calculate the Fitness Function

The fitness function was used as the performance evaluation of particles. Each particle’s fitness function was expressed by the total makespan, the duration of the whole multi-project schedule. Each particle in the population was evaluated and the local best of each was recorded. The best particle in the population was selected as the global best.

#### 3.2.3. Evolve Approach

The crossover operator means that the following generations inherit the good genes from the previous generation. The crossover operator generates new solutions (offspring) by coalescing the structures of a couple of existing ones (parents) [30]. For our specific representation, the crossover technique defined in the literature was not suitable. We considered the task list concept, precedence feasibility, and the mode assignment. We used the well-known one-point crossover within the TL sector and the two-point crossover within the mode assignment sector. The crossover was applied to each particle with the selected local best and global best. Each new generated particle was evaluated and the best particle was selected as the new global best particle. The local best for each particle was recorded. The one-point crossover and two-point crossover are described as follows:

Given two individuals selected for crossover,  $M = (j_1^M, j_2^M, \dots, j_j^M)$  and  $F = (j_1^F, j_2^F, \dots, j_j^F)$ , two offspring,  $D$  and  $S$  are produced.

##### 1. One-point crossover operator

First, one integer and non-negative point  $p(1 < p < J)$  are randomly generated. The tasks in the first  $p$  positions are inherited from the  $M$ , exactly in the same sequence.

$$j_i^D := j_i^M \quad i = 1, \dots, p \tag{12}$$

The positions  $i = p + 1, \dots, J$  are inherited from the  $F$  and preserved their relative sequence.

$$j_i^D := j_k^F \quad i = p + 1, \dots, J \tag{13}$$

$$k = \min \left\{ k \mid j_k^F \notin \{j_1^D, \dots, j_{i-1}^D\}, k = 1, 2, \dots, J \right\} \tag{14}$$

The  $S$  is generated analogously and the sequence of succession is opposite.

$$j_i^S := j_i^F \quad i = 1, \dots, p \tag{15}$$

$$j_i^S := j_k^M \quad i = p + 1, \dots, J \tag{16}$$

$$k = \min \left\{ k \mid j_k^M \notin \{j_1^S, \dots, j_{i-1}^S\}, k = 1, 2, \dots, J \right\} \tag{17}$$

## 2. Two-point crossover operator

First, two integers and non-negative points  $p_1, p_2 (1 < p_1 < p_2 < J)$  are randomly generated. The tasks in the first  $p_1$  positions are inherited from the  $M$ , exactly in the same sequence.

$$j_i^D := j_i^M \quad i = 1, \dots, p_1 \tag{18}$$

The positions  $i = p_1 + 1, \dots, p_2$  are inherited from the  $F$  and preserved their relative sequence.

$$j_i^D := j_k^F \quad i = p_1 + 1, \dots, p_2 \tag{19}$$

$$k = \min \left\{ k \mid j_k^F \notin \{j_1^D, \dots, j_{i-1}^D\}, k = 1, 2, \dots, J \right\} \tag{20}$$

The positions  $i = p_2 + 1, \dots, J$  are inherited from the  $M$  again and preserved their relative sequence.

$$j_i^D := j_k^M \quad i = p_2 + 1, \dots, J \tag{21}$$

$$k = \min \left\{ k \mid j_k^M \notin \{j_1^D, \dots, j_{i-1}^D\}, k = 1, 2, \dots, J \right\} \tag{22}$$

The  $S$  is generated analogously and the sequence of succession is opposite.

$$j_i^S := j_i^F \quad i = 1, \dots, p_1 \tag{23}$$

$$j_i^S := j_k^M \quad i = p_1 + 1, \dots, p_2 \tag{24}$$

$$k = \min \left\{ k \mid j_k^M \notin \{j_1^S, \dots, j_{i-1}^S\}, k = 1, 2, \dots, J \right\} \tag{25}$$

$$j_i^S := j_k^F \quad i = p_2 + 1, \dots, J \tag{26}$$

$$k = \min \left\{ k \mid j_k^F \notin \{j_1^S, \dots, j_{i-1}^S\}, k = 1, 2, \dots, J \right\} \tag{27}$$

If the termination criterion were satisfied, usually a sufficiently good fitness or a special number of generations, then the PSO phase was stopped; otherwise it continued. The PSO phase could provide diverse initial solutions for the TS phase.

### 3.3. TS Phase

#### 3.3.1. Initial Population

In our algorithm, the initialization solution of TS was provided by the improved PSO. Individuals of the best population were randomly selected, making sure that the global best solution was included. The solutions were labeled and the following steps were performed in sequence.

#### 3.3.2. Generate the Neighborhood Solution

The neighborhood structure plays a very important role in a local search. The neighborhood structure is a mechanism which can apply a small perturbation to the given solution in order to obtain a new set of neighboring solutions. The neighborhood structure is directly effective on the efficiency of the local search, and unnecessary and infeasible moves must be eliminated if it is possible. In this study, the best neighborhood which was non-Tabu or satisfied the aspiration criterion was selected.

Three different neighborhood operators were implemented:

(1) A neighborhood shift in which operators only performed the list of tasks in the following way:

One task  $j$  was randomly chosen from the list of tasks; the nearest predecessor  $p$ . of task  $j$  and the nearest successor  $s$ . of task  $j$  were found in the task list; a position  $x$  between tasks  $p$  and  $s$  was randomly chosen; task  $j$  was moved to position  $x$  and all jobs between job  $j$  and job in position  $x + 1(x - 1)$  were shifted to the left (or to the right). We show an example in Figure 6. Task 5 was randomly chosen, its nearest predecessor is task 1, its nearest successor is task 6. Position 5 of the

TL sector was the position of task 4, then task 5 was moved to position 5, and task 2 and task 4 were shifted to the left.

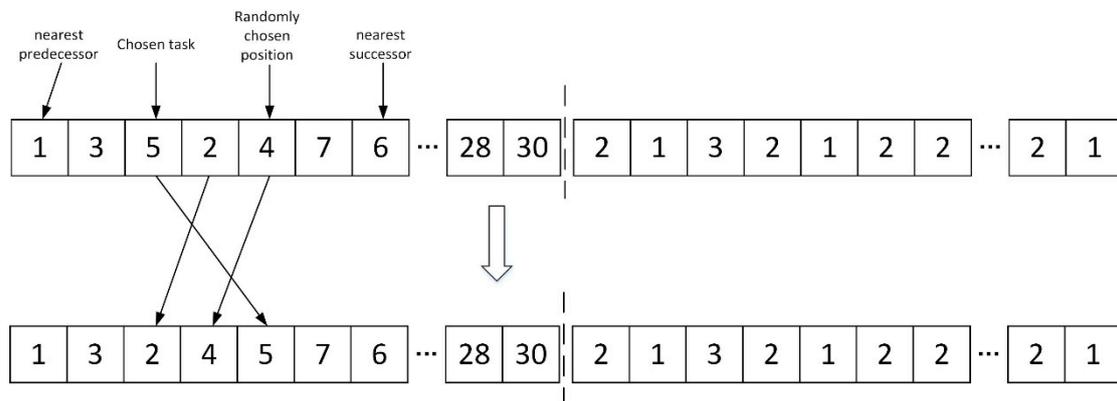


Figure 6. An example of the neighborhood shift operator.

(2) A mode change which operated only on the mode assignment in the following way:

One task  $j$  was randomly chosen from the list of tasks, the corresponding mode of task  $j$  was changed to another randomly chosen one, if possible. We show an example in Figure 7. Task 7 was randomly chosen, its corresponding mode was 2, then changed to mode 1.

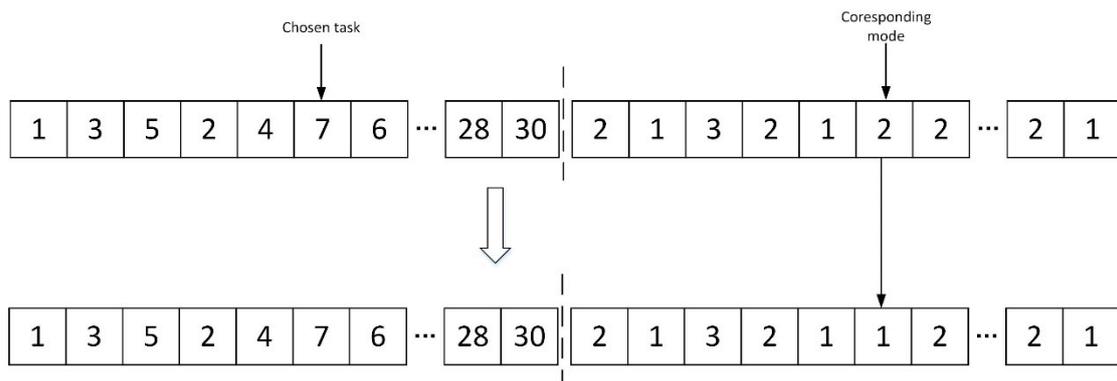


Figure 7. An example of the mode change operator.

(3) A combined move which operated simultaneously on both the vectors and was a composition of the swap and the mode change. We show an example in Figure 8.

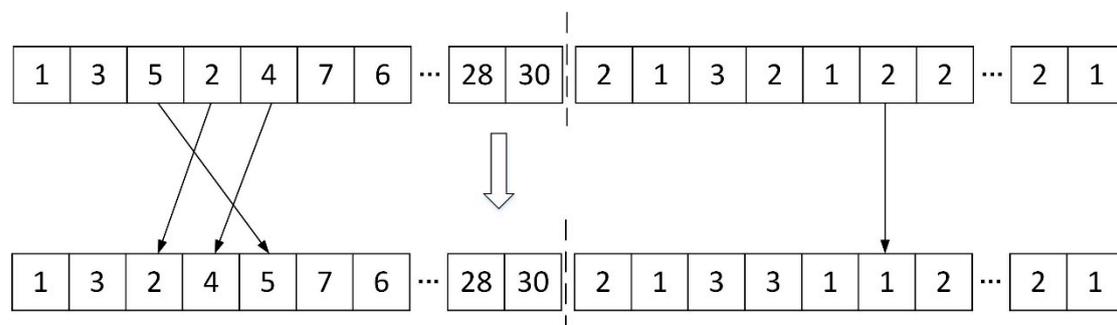


Figure 8. An example of a combined move operator.

Each neighborhood was repeatedly searched until no improvement was identified. Then, the next neighborhood operator was taken. The Tabu list was updated and the termination criterion were checked.

### 3.3.3. Aspiration Criterion

The aspiration rule was introduced to release the Tabu list. It avoided the deadlock when all solutions were prohibited in the Tabu list, or when a neighborhood optimal solution was within the Tabu list but was better than the current optimal solution. The aspiration rule utilized this solution and was not subject to Tabu constraints.

### 3.3.4. The Tabu List

In order to avoid getting stuck in the cycle state of the local optima during searching, some steps were prohibited from repeating. The idea of the Tabu list was introduced, where the position of a move and the solution were stored in the Tabu list. When a new neighboring solution was found or an old solution expired, the Tabu list was updated.

### 3.3.5. Termination Criterion

The termination criterion was used to determine whether the proposed method should stop. If one of the following conditions is satisfied; whether the proposed method stops when the algorithm has performed a given total number of iterations, or the elite solution stack had been exhausted, or the solution was proven to be optimal.

## 4. Case Study

In this section, we present the results of computational studies to evaluate the performance of the proposed algorithm. Our algorithm was coded in MATLAB, and the software version was MATLAB R2016a, running on a personal computer configured with 16 GB of memory and an Intel Core i7-6700 3.4GHz processor.

### 4.1. Implementation Details

Instances in this paper came from the MISTA 2013 Challenge, which combined multiple MRCPSP instances from Project Scheduling Problem Library (PSPLIB). Release dates had to be added and the release dates were generated using a Poisson process, i.e., the project arrival times were exponentially distributed with  $\lambda = 0.2$ . The release date of the first project was always  $r_0 = 0$ . The global resource was ignored and the two renewable resources were regarded as two hierarchical levels of workers. The instances consisted of three datasets from three different competition phases, and each dataset consisted of ten instances. The maximum number of tasks for each project that could be executed concurrently was set at 2. Problem size in the first dataset, which was called A instances, ranged from 2 projects and 10 tasks per project to 10 projects and 30 tasks per project. The second dataset called B instances varied from 10 projects and 10 tasks per project to 20 projects and 30 tasks per project in problem size. The last dataset was called X instances, with instances of similar size as the B instances [19].

### 4.2. Performance Evaluation of the Proposed Algorithm

To evaluate the proposed algorithm, comparisons were made with improved particle swarm optimization (PSO) proposed by Jia and Seo [31], the Tabu search (TS) proposed by Pan et al., [32], and the genetic algorithm (GA) proposed by Goncharov and Leonov [33]. The parameters of GA, PSO, and TS were set by trial and error. The scheduling results of PSO-TS, GA, PSO, and TS are shown in Table 1, including the best makespan (BST) and the average makespan (AVG) of ten independent runs. The best BST results of each instance are set in bold. From Table 1, it is shown that the BST and AVG values

obtained by PSO–TS were better than those obtained by the other three algorithms on all instances, which demonstrated that PSO–TS had the superior searching quality and robustness. In most of the instances, GA performed better than PSO and TS, but it was not consistent. In instance A7, the BST and AVG of TS performed better than GA. In instance X7, the BST and AVG of PSO performed better than GA.

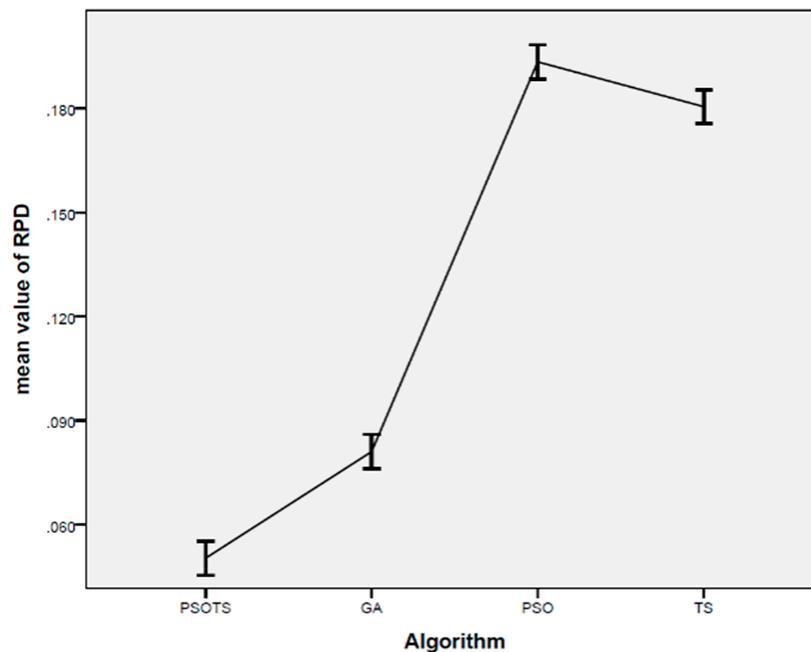
**Table 1.** Comparisons of Particle swarm optimization-Tabu search (PSO–TS), Genetic algorithm (GA), Particle swarm optimization (PSO), and Tabu search (TS) over ten runs.

Instances	GA		PSO		TS		PSO–TS	
	AVG	BST	AVG	BST	AVG	BST	AVG	BST
A1	50.1	49	57.3	50	54.6	51	49.9	<b>48</b>
A2	89	85	101.9	94	104.2	97	88	<b>84</b>
A3	119.2	116	131.8	123	144.4	137	115.4	<b>112</b>
A4	115.2	108	134.3	128	125.6	111	112.2	<b>107</b>
A5	293.4	275	339.5	313	307.8	289	281.1	<b>258</b>
A6	378.4	360	451.8	445	377.7	353	372.5	<b>345</b>
A7	352.4	341	400.6	367	352.2	338	343.3	<b>332</b>
A8	310.4	305	341.5	323	358.3	346	298.4	<b>292</b>
A9	457.1	436	495.1	476	496	475	451.9	<b>422</b>
A10	515.7	501	549.3	529	556.9	539	483.1	<b>449</b>
B1	186.9	176	221.1	202	222.5	213	177.5	<b>169</b>
B2	358.3	343	398.8	379	402	383	348	<b>330</b>
B3	801.8	779	889.7	864	824.8	800	786.1	<b>740</b>
B4	370.3	357	416.1	396	403	380	356.7	<b>340</b>
B5	566.4	551	625.9	607	630.7	606	545.2	<b>522</b>
B6	701.2	675	744.9	720	763	743	666	<b>630</b>
B7	400	384	448	436	439.8	412	376.9	<b>362</b>
B8	743.5	706	823.3	795	834.3	796	705.8	<b>679</b>
B9	1397	1352	1441.8	1367	1430.9	1355	1384.7	<b>1325</b>
B10	944.9	912	996.9	932	1016.1	922	944.3	<b>886</b>
X1	183.6	172	215.1	206	215.1	202	178	<b>170</b>
X2	299.9	293	336.5	318	347.3	331	291.9	<b>279</b>
X3	496.1	473	529.4	502	540.8	523	469.8	<b>450</b>
X4	215.2	209	240	228	252.9	237	205	<b>196</b>
X5	594.1	568	659.5	616	647.2	625	574.1	<b>543</b>
X6	842.5	811	917.2	893	905.7	883	798.1	<b>765</b>
X7	378.2	361	375.8	352	413.3	377	367.9	<b>350</b>
X8	1058.2	1021	1095.2	1032	1071.2	1021	1056.4	<b>996</b>
X9	2025.3	1950	2028.8	1891	1982.2	1942	1952	<b>1878</b>
X10	1342	1289	1390.7	1295	1357.5	1284	1326.7	<b>1275</b>

To compare results obtained in different instances, relative percentage deviation (RPD) was introduced as the only dependent variable of the variance analysis, as shown in equation (28), where  $Alg_{sol}$  represents the objective value obtained by a single algorithm running, and  $BST_{sol}$  represents the best solution over the whole set of results concerning the same instance. Obviously, the smaller the RPD value, the better the result.

$$RPD = \frac{Alg_{sol} - BST_{sol}}{BST_{sol}} \times 100 \tag{28}$$

The ANOVA and least-significant difference (LSD) tests were conducted in SPSS to check the results transformed into the RPD value. Test results revealed that under a confidence interval of 95%, the  $p$  value was 0, which means there were significant differences in the performance of the four algorithms. Figure 9 depicts the mean plot with the LSD intervals for the RPD value obtained by the four algorithms. In this measure, the proposed algorithm outperformed the other three algorithms.



**Figure 9.** Mean plot and Least-significant (LSD) intervals for algorithms in relative percentage deviation (RPD) value.

## 5. Industrial Application

To demonstrate the applicability of the proposed method from a practical point of view, a case that uses industrial data from a collaborating steam turbine company was considered. The steam turbine is typically an engineer-to-order product [34]. We collaborated with the Shanghai Turbine Plant (STP), which is the leading turbine company in China. STP can provide an accurate estimation of task duration based on the number of allocated workers during the engineering process. They have established the relationship between task duration and number of allocated workers based on their experience. The steam turbine assembly shop had eight work centers, and eight steam turbines could be assembled in parallel. There were eight kinds of steam turbine products to be produced, and the total number of tasks was 270. The assembly of each product was rather different in terms of tasks, due date, precedence constraints, and number of workers involved. Different projects were simultaneously executed, competing for and sharing the worker resources. Therefore, it is necessary to schedule and balance the utilization of worker resources to guarantee the due date of all products. Figure 10 depicts the sequence diagram of eight projects. The assembly shop had two skill levels of workers—nine lower level workers (junior) and six higher level workers (senior). Workers formed the assembly team to complete the task. The release dates and due dates of all products were determined by a higher level production plan. We set up the real-time visual monitoring system for steam turbine assembly and the system collected real-time assembly information. The project manager operating the real-time visual monitoring system is shown in Figure 11 and the user interface is shown in Figure 12.

We ran our algorithm for the above case on ten independent runs and selected the best result. The Gantt chart of the GA algorithm can be seen in Figure 13a, in which the makespan was 1630 h. The Gantt chart of the PSO algorithm can be seen in Figure 13b, in which the makespan was 1678 h. The Gantt chart of the TS algorithm can be seen in Figure 13c, in which the makespan was 1642 h. The Gantt chart of the plan made by the project manager can be seen in Figure 13d, in which the makespan was 1808. The Gantt chart of the PSO–TS algorithm can be seen in Figure 14, in which the makespan was 1524 h. In Figure 14, the horizontal axis represents the time horizon, the vertical axis represents the project number, the box represents the task, the number in the box represents the task number, and the length represents the duration of the task. We saw the sequence and the duration of tasks in each

project, and found that some tasks could be executed concurrently under the space constraints. The comparison of these five project plans can be seen in Figure 15. The cylinder in the figure represents the makespan, a larger cylinder means that the corresponding makespan was longer. It is clear that the PSO-TS algorithm outperformed the solutions provided by the other three algorithms and the experienced manager. The PSO-TS algorithm we proposed can assist in scheduling the assembly process of steam turbines.

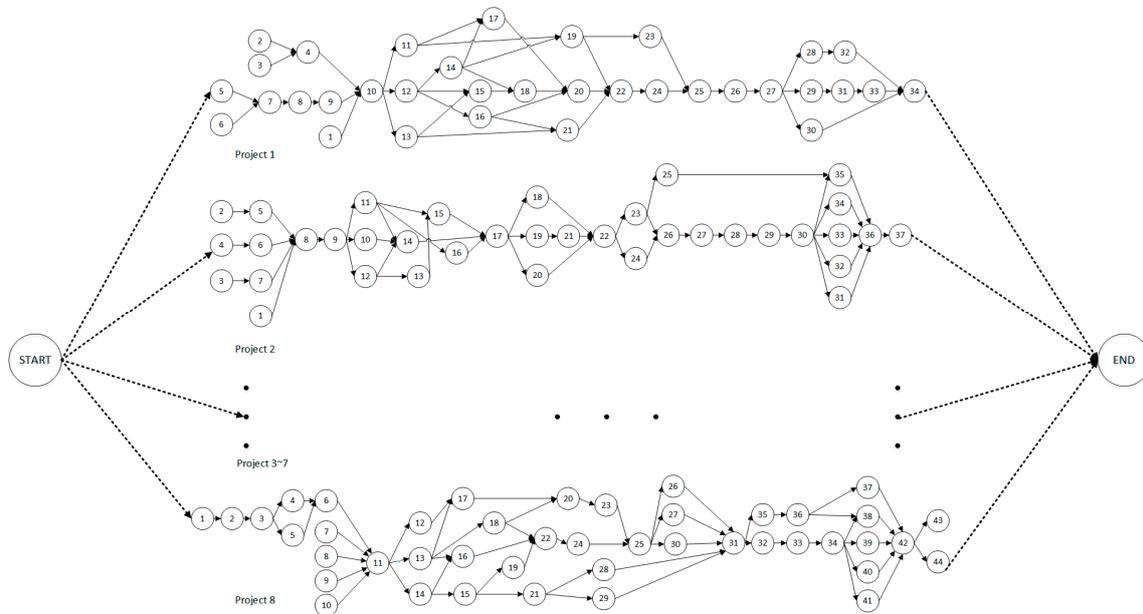


Figure 10. Sequence diagram of eight projects.

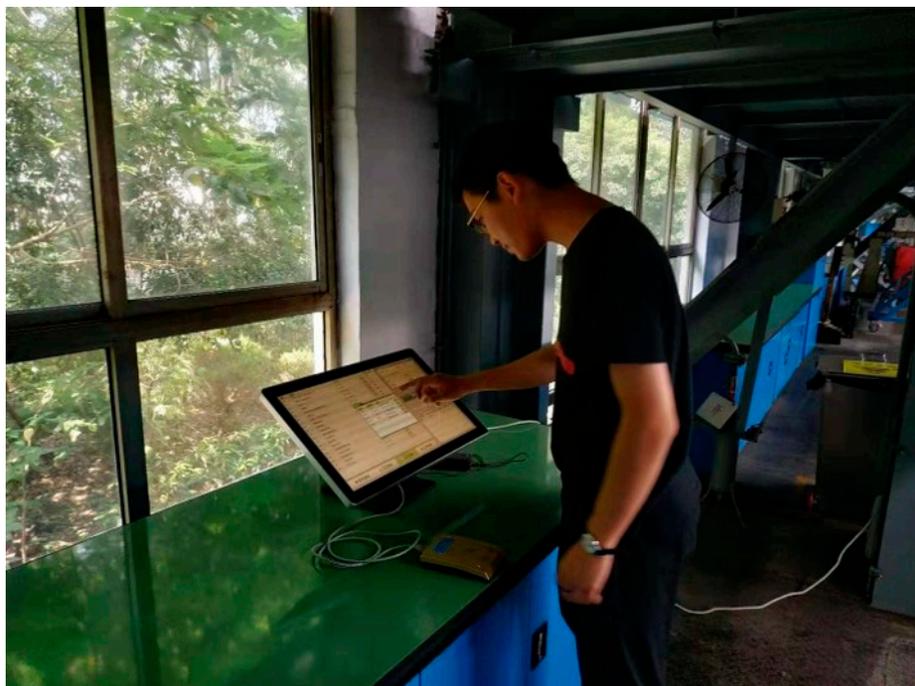


Figure 11. The project manager operating the real-time visual monitoring system.

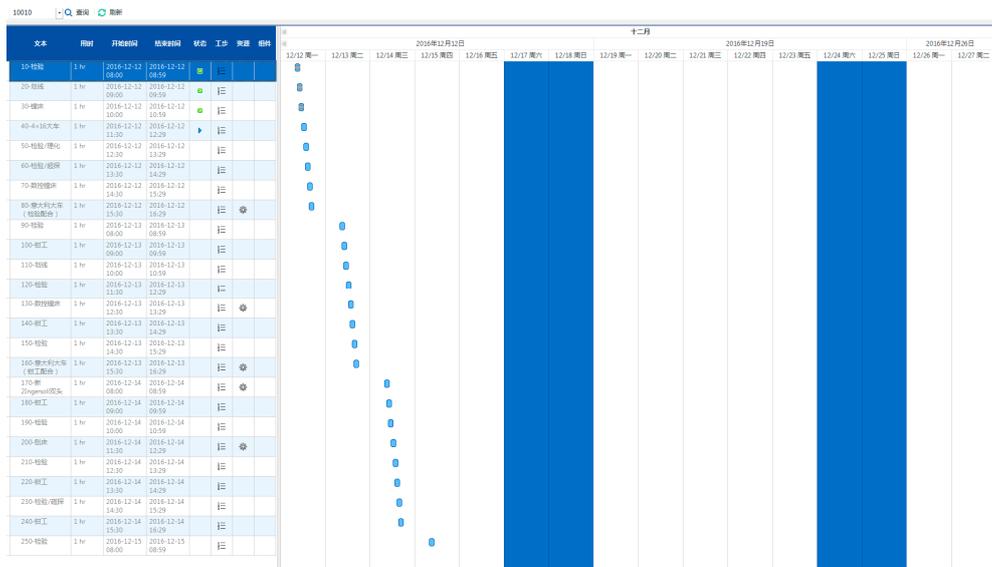


Figure 12. The user interface of the real-time visual monitoring system.

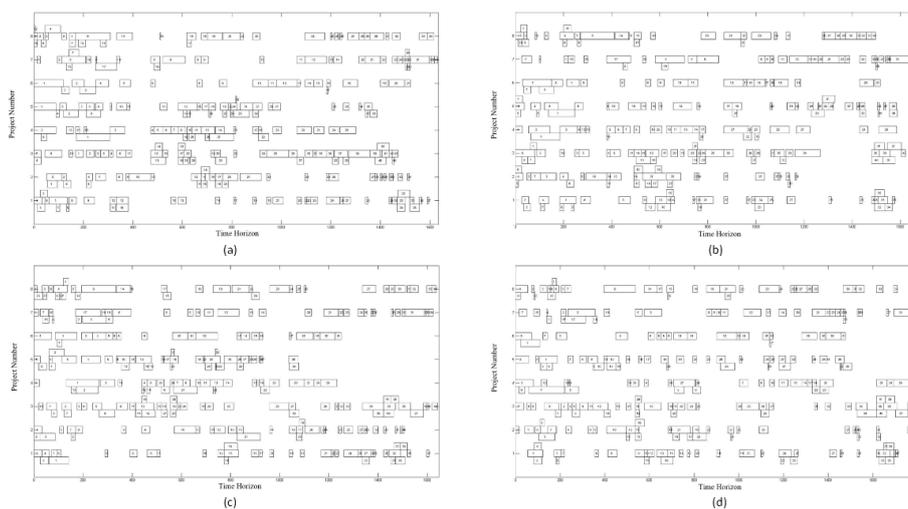


Figure 13. Gantt chart of plans provided by Genetic algorithm (GA), Particle swarm optimization (PSO), Tabu search (TS), and the project manager.

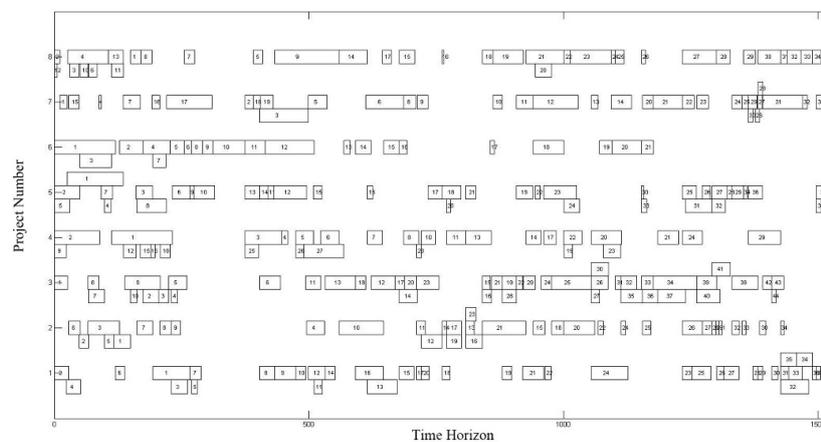


Figure 14. Gantt chart of the Particle swarm optimization-Tabu search (PSO-TS) algorithm.

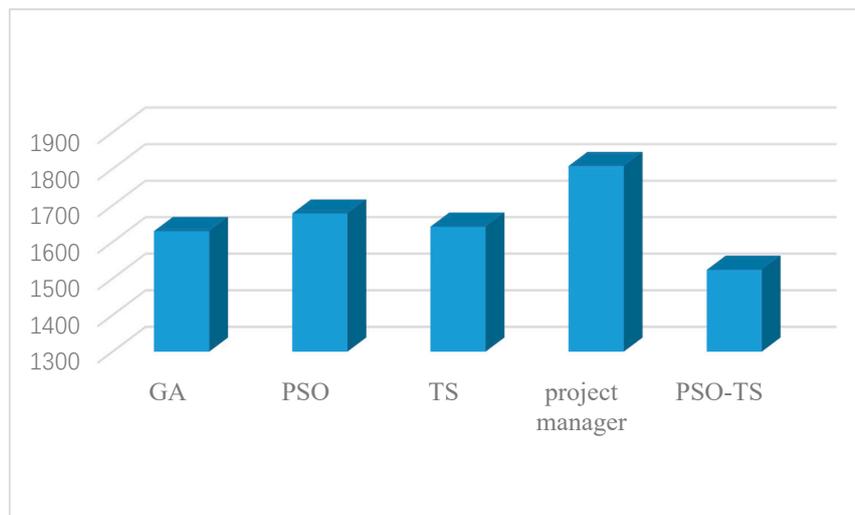


Figure 15. Comparison of the five project plans.

## 6. Conclusions

In this paper, we present a mathematical model for integrated multi-project scheduling and hierarchical workforce allocation in the ETO assembly process. A hybrid approach combining improved particle swarm optimization and Tabu search was proposed to solve the problem. The proposed algorithm was tested on different scale benchmark instances and a case that used industrial data from a collaborating ETO company. The computational results show the superiority of the hybrid algorithm. The hybrid algorithm was applied to assist the scheduling system, and it not only reduced the workload of the project manager, but also provided a better scheduling plan. The main contribution of this research was the focus on project scheduling and workforce allocation in the ETO assembly process and it can expand the research content in this field.

Directions for future research can be outlined as follows: Firstly, some heavy and large components needed to be transported by crane, thus the project scheduling and crane scheduling should be dealt with together. Secondly, the problem we studied did not take unexpected events into account, for instance the deviation of the actual duration from the planned duration, the late delivery of necessary components, or rework due to quality problems. These disruptions will delay the finish time of tasks and affect other projects. Future work is needed to apply the hybrid algorithm to dynamic scheduling in the ETO assembly process.

**Author Contributions:** C.J. and X.H. proposed the method and wrote the paper; J.X. conceived and designed the experiments; and C.J. and J.X. performed the experiments and analyzed the data.

**Acknowledgments:** The authors would like to acknowledge the financial support of the National Natural Science Foundation of China (No. 51475303).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Strandhagen, J.W.; Vallandingham, L.R.; Alfnes, E.; Strandhagen, J.O. Operationalizing lean principles for lead time reduction in engineer-to-order (ETO) operations: A case study. *IFAC-PapersOnLine* **2018**, *51*, 128–133. [[CrossRef](#)]
2. Grabenstetter, D.H.; Usher, J.M. Sequencing jobs in an engineer-to-order engineering environment. *Prod. Manuf. Res.* **2015**, *3*, 201–217. [[CrossRef](#)]
3. Hicks, C.; Song, D.P.; Earl, C.F. Dynamic scheduling for complex engineer-to-order products. *Int. J. Prod. Res.* **2007**, *45*, 3477–3503. [[CrossRef](#)]

4. Wang, Z.; Zhan, D.; Xu, X. Service-Oriented Infrastructure for Collaborative Product Design in ETO Enterprises. In Proceedings of the International Conference on Computer Supported Cooperative Work in Design, Nanjing, China, 3–5 May 2006; pp. 1–6.
5. Grabenstetter, D.H.; Usher, J.M. Developing due dates in an engineer-to-order engineering environment. *Int. J. Prod. Res.* **2014**, *52*, 6349–6361. [[CrossRef](#)]
6. Alfieri, A.; Tolio, T.; Urgo, M. A project scheduling approach to production and material requirement planning in Manufacturing-to-Order environments. *J. Intell. Manuf.* **2012**, *23*, 575–585. [[CrossRef](#)]
7. De Lit, P.; Latinne, P.; Rekiek, B.; Delchambre, A. Assembly planning with an ordering genetic algorithm. *Int. J. Prod. Res.* **2001**, *39*, 3623–3640. [[CrossRef](#)]
8. Alfieri, A.; Tolio, T.; Urgo, M. A two-stage stochastic programming project scheduling approach to production planning. *Int. J. Adv. Manuf. Technol.* **2012**, *62*, 279–290. [[CrossRef](#)]
9. Oztemel, E.; Selam, A.A. Bees Algorithm for multi-mode, resource-constrained project scheduling in molding industry. *Comput. Ind. Eng.* **2017**, *112*, 187–196. [[CrossRef](#)]
10. Sungur, B.; Yavuz, Y. Assembly line balancing with hierarchical worker assignment. *J. Manuf. Syst.* **2015**, *37*, 290–298. [[CrossRef](#)]
11. Hytonen, J.; Niemi, E.; Toivonen, V. Optimal workforce allocation for assembly lines for highly customised low-volume products. *Int. J. Serv. Oper. Inf.* **2008**, *3*. [[CrossRef](#)]
12. Salido, M.A. Introduction to planning, scheduling and constraint satisfaction. *J. Intell. Manuf.* **2010**, *21*, 1–4. [[CrossRef](#)]
13. Adrodegari, F.; Bacchetti, A.; Pinto, R.; Pirola, F.; Zanardini, M. Engineer-to-order (ETO) production planning and control: An empirical framework for machinery-building companies. *Prod. Plan. Control* **2015**, *26*, 910–932. [[CrossRef](#)]
14. Heimerl, C.; Kolisch, R. Scheduling and staffing multiple projects with a multi-skilled workforce. *OR Spectrum* **2010**, *32*, 343–368. [[CrossRef](#)]
15. Mencía, R.; Sierra, M.R.; Mencía, C.; Varela, R. Genetic algorithms for the scheduling problem with arbitrary precedence relations and skilled operators. *Integr. Comput. Aided Eng.* **2016**, *23*, 269–285. [[CrossRef](#)]
16. Karam, A.; Attia, E.A.; Duquenne, P. A MILP model for an integrated project scheduling and multi-skilled workforce allocation with flexible working hours. *IFAC-PapersOnLine* **2017**, *50*, 13964–13969. [[CrossRef](#)]
17. Lian, J.; Liu, C.G.; Li, W.J.; Yin, Y. Multi-skilled worker assignment in seru production systems considering worker heterogeneity. *Comput. Ind. Eng.* **2018**, *118*, 366–382. [[CrossRef](#)]
18. Yuan, P.L.; Han, W.; Su, X.C.; Liu, J.; Song, J.Y. A Dynamic Scheduling Method for Carrier Aircraft Support Operation under Uncertain Conditions Based on Rolling Horizon Strategy. *Appl. Sci.-Basel* **2018**, *8*. [[CrossRef](#)]
19. Wauters, T.; Kinable, J.; Smet, P.; Vancroonenburg, W.; Berghe, G.V.; Verstichel, J. The Multi-Mode Resource-Constrained Multi-Project Scheduling Problem. *J. Sched.* **2016**, *19*, 1–13. [[CrossRef](#)]
20. Geiger, M.J. A multi-threaded local search algorithm and computer implementation for the multi-mode, resource-constrained multi-project scheduling problem. *Eur. J. Oper. Res.* **2017**, *256*, 729–741. [[CrossRef](#)]
21. Toffolo, T.A.; Santos, H.G.; Carvalho, M.A.; Soares, J.A. An integer programming approach to the multimode resource-constrained multiproject scheduling problem. *J. Sched.* **2016**, *19*, 295–307. [[CrossRef](#)]
22. Alonso-Pecina, F.; Pecero, J.; Romero, D. A three-phases based algorithm for the multi-mode resource-constrained multi-project scheduling problem. In Proceedings of the 6th Multidisciplinary International Scheduling Conference, Gent, Belgium, 27–29 August 2010; pp. 812–814.
23. Sonmez, R.; Uysal, F. Backward-Forward Hybrid Genetic Algorithm for Resource-Constrained Multiproject Scheduling Problem. *J. Comput. Civ. Eng.* **2014**, *29*, 04014072. [[CrossRef](#)]
24. Gao, L.; Li, X.; Wen, X.; Lu, C.; Wen, F. A hybrid algorithm based on a new neighborhood structure evaluation method for job shop scheduling problem. *Comput. Ind. Eng.* **2015**, *88*, 417–429. [[CrossRef](#)]
25. Feng, H.; Da, W.; Xi, L.; Pan, E.; Xia, T. Solving the integrated cell formation and worker assignment problem using particle swarm optimization and linear programming. *Comput. Ind. Eng.* **2017**, *110*, 126–137. [[CrossRef](#)]
26. Gen, M.; Lin, L. Multiobjective evolutionary algorithm for manufacturing scheduling problems: State-of-the-art survey. *J. Intell. Manuf.* **2014**, *5*, 849–866. [[CrossRef](#)]
27. Eberhart, R.; Kennedy, J. New optimizer using particle swarm theory. In Proceedings of the 1995 6th International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.

28. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
29. Glover, F. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* **1986**, *13*, 533–549. [[CrossRef](#)]
30. Hartmann, S. Competitive genetic algorithm for resource-constrained project scheduling. *Nav. Res. Log.* **1998**, *45*, 733–750. [[CrossRef](#)]
31. Jia, Q.; Seo, Y. An improved particle swarm optimization for the resource-constrained project scheduling problem. *Int. J. Adv. Manuf. Technol.* **2013**, *67*, 2627–2638. [[CrossRef](#)]
32. Pan, N.-H.; Hsaio, P.-W.; Chen, K.-Y. A study of project scheduling optimization using Tabu Search algorithm. *Eng. Appl. Artif. Intell.* **2008**, *21*, 1101–1112. [[CrossRef](#)]
33. Goncharov, E.N.; Leonov, V.V. Genetic algorithm for the resource-constrained project scheduling problem. *Autom. Remote Control* **2017**, *78*, 1101–1114. [[CrossRef](#)]
34. Yang, Q.H.; Qi, G.N.; Lu, Y.J.; Gu, X.J. Applying mass customization to the production of industrial steam turbines. *Int. J. Comput. Integr. Manuf.* **2007**, *20*, 178–188. [[CrossRef](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).