infrastructures

MDPI

Article

A Timed Petri Net Method to Optimize the Scheduling of a Railway Hub Construction Project

Wei Wang * and Enjian Yao

School of Traffic and Transportation, Beijing Jiaotong University, Beijing 100044, China
* Correspondence: 20114076@bjtu.edu.cn

Abstract

The construction of large-scale buildings often faces extended production cycles due to inefficiencies in scheduling processes. To address this challenge, a timed Petri net model was developed to analyze and optimize construction scheduling. Based on the Petri net transition sequence, a scheduling optimization model was proposed. To solve the model efficiently, an improved brainstorming optimization (BSO) algorithm was introduced. Compared with the classical BSO, two targeted enhancements were introduced: a problem-specific encoding and decoding method for Petri net transition sequences to ensure solution feasibility and an embedded simulated annealing local search mechanism to prevent premature convergence in later iterations. The proposed methodology was validated using real-world data from a large high-speed railway hub foundation pit construction project. Results demonstrated a significant reduction of 531 working hours in the total scheduling time, representing a 15.47% improvement in scheduling efficiency compared to traditional sequential scheduling methods. This approach not only shortened the scheduling cycle and enhanced production efficiency but also offered an innovative solution to address scheduling issues in complex construction processes.

Keywords: timed Petri net; BSO algorithm; scheduling

1. Introduction

Optimizing building construction processes in large-scale projects is essential for improving efficiency, reducing costs, and maintaining safety while adhering to tight deadlines and quality standards. Construction project optimization (CPO) plays a critical role in both manufacturing and service industries, as it directly impacts operational efficiency and overall organizational performance. Fundamentally, CPO involves resource allocation and scheduling tasks, which can be characterized as a job shop scheduling problem (JSP). In construction projects, multiple materials must be processed in a specific order according to distinct construction techniques, with each step requiring specific resources and time. These characteristics align closely with the JSP, allowing the problem to be addressed using similar optimization techniques [1]. This challenge is especially prominent in railway construction projects, where high-speed railway stations and hubs involve highly complex multi-process workflows subject to strict safety, time, and resource constraints.

Over the years, CPO has drawn significant attention from academia, leading to the development of various scheduling methods. Traditional approaches, such as mathematical programming and heuristic scheduling [2–4], have been widely applied. However, the inherent flexibility of manufacturing paths in such problems often exceeds the capabilities of these traditional methods. From a control perspective, CPO represents a complex discrete

Check for updates

Academic Editor: Giuseppe Cantisani

Received: 17 June 2026
Revised: 10 August 2026
Accepted: 10 August 2026
Published: 25 August 2026

Copyright: © 2026 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.

event system (DES), where system states change at discrete time points triggered by specific events. Petri nets (PNs) have emerged as a robust tool for modeling, analyzing, and simulating DESs due to their logical, graphical representation and mathematical analytical capabilities [5,6]. By systematically modeling production line constraints and features, PNs enable the derivation of optimal solutions through optimization algorithms. This makes it possible to prepare efficient scheduling plans, laying a solid foundation for rapid production organization and cost reduction.

The motivation of this study stems from two observed gaps in the existing literature. First, current PN-based scheduling methods rely heavily on GA or PSO, both of which struggle with feasibility and premature convergence in large-scale construction scheduling. Second, standard BSO, while diversity-preserving, lacks a mechanism to escape local optima in the later search stages, which limits its practical effectiveness in complex PN-based problems.

The novelty of this work lies in two aspects. First, a problem-specific encoding and decoding method is designed for PN transition sequences, which guarantees solution feasibility by construction and eliminates the need for repair procedures. Second, an SA acceptance mechanism is embedded directly into the BSO update process, enabling principled escape from local optima while preserving the cluster-based diversity that BSO relies on for global search. The contributions of this paper are as follows.

(1) A timed Petri net model is established for multi-batch construction scheduling, and the scheduling problem is formulated as a transition sequence optimization problem with minimization of total construction period as the objective.

(2) The improved BSO algorithm is developed and validated on a real high-speed railway hub construction project, demonstrating a significant reduction in total scheduling time compared with sequential scheduling.

The remainder of this paper is organized as follows. Section 3 describes the proposed BSO-based optimization algorithm. Section 4 presents the mathematical scheduling model. Section 5 presents a case study. Section 6 concludes the paper.

2. Literature Review

In recent years, PN-based research has expanded to cover system modeling, analysis, and control in DESs. For instance, Schild proposed an extended Petri net formalism as a suitable language for industrial production processes, capturing the dynamics and constraints of entire systems [7]. Zhou applied timed Petri nets to model batch production systems [8], while Liu et al. utilized object-oriented extended-time Petri nets for simulating and analyzing manufacturing systems, improving real-time scheduling and execution [9]. The scheduling optimization problem in PNs is equivalent to solving the optimal trigger sequence of the PN [10]. Thus, heuristic functions play an important role to the enhancement of scheduling speed and quality [11]. However, as the singleness of traditional manufacturing can no longer meet market demand, orders tend to be small batches and multiple batches, and the constructed model involves more variables and constraints [12,13]. Therefore, when solving the model, existing research has increasingly focused on combining PNs with swarm intelligence (SI) optimization algorithms to address complex scheduling problems in real industrial settings.

Despite this progress, most existing PN-based hybrid scheduling methods rely on GA or PSO as the core optimizer. When encoding PN transition sequences within these frameworks, both GA and PSO frequently generate infeasible firing sequences that violate token-flow constraints, requiring costly repair procedures. More importantly, in large-scale construction scheduling scenarios such as high-speed railway station construction, the solution space grows combinatorially with the number of processes and resources. Under

such conditions, GA tends to lose population diversity prematurely, while PSO struggles to maintain the balance between global exploration and local exploitation. These factors make it difficult for existing PN-based hybrid methods to reliably find near-optimal scheduling plans within a reasonable computation time. The present work is motivated by these observations, and it differs from prior PN-based approaches in two key respects. First, BSO is adopted as the optimizer because its cluster-based structure naturally maintains solution diversity and captures collective knowledge across different regions of the search space, which is more compatible with the complex feasibility constraints encoded in PN transition sequences. Second, rather than simply replacing the optimizer, we integrate an SA acceptance mechanism directly into the BSO update process, giving the algorithm a principled way to escape local optima without discarding the clustering structure that drives BSO's global search capability.

SI is a novel approach that simulates the biological species' collective behaviors and consists of a variety of different algorithms. These algorithms have the advantages of robustness and outstanding global searching capability, which are appropriate for complex problems solving. Among these algorithms, genetic algorithm (GA), particle swarm optimization (PSO), and simulated annealing algorithm (SA) are widely utilized [14–16]. For example, Dai proposed a hybrid approach that combines GA and SA to address CPO in reducer production projects [17]. Dao applied GA to address CPO problems in virtual computer-integrated manufacturing systems [18]. Anuar proposed discrete PSO to solve CPO by minimizing the production cycle [19]. Rahman uses PSO to solve the problem of real-time scheduling in order-oriented workshop production systems [20]. Lim devised an SA-based hyper-heuristic to solve the flexible job shop scheduling problem [21]. Although the three algorithms are widely used for production line scheduling, they each have notable limitations. GA often suffers from premature convergence due to a loss of genetic diversity and exhibits slow convergence speed, especially in complex problems [21,22]. PSO faces challenges in balancing exploration and exploitation, leading to premature convergence or long computation times, and its performance degrades with increasing problem size and complexity [23,24]. SA, while designed to escape local optima, is computationally expensive and highly sensitive to the cooling schedule, making it difficult to tune for complex problems [25,26]. Thus, these algorithms exhibit two common limitations, namely, local optimum and low computational efficiency [27,28].

To solve the two limitations, we adopt the brainstorming optimization (BSO) algorithm, a promising SI algorithm originally presented by Shi in 2011, and improve it for the construction scheduling problem addressed in this paper [29]. It simulates the collective behavior of human beings, namely, the process of brainstorming [30]. BSO leverages expert knowledge and experience to circumvent local optima, enhancing the probability of identifying global optima [31–33].

To motivate the proposed improvement more concretely, consider a simplified construction scheduling instance involving three sequential processes, namely, foundation work, structural erection, and finishing, which must be completed across two parallel work zones under shared equipment and labor constraints. When the standard BSO is applied to this instance, individuals within each cluster tend to converge to a narrow set of PN transition firing sequences after only a moderate number of iterations. Because BSO updates rely primarily on intra-cluster and inter-cluster information exchange, once the clusters lose diversity, the algorithm can no longer generate meaningfully different firing sequences, and the makespan curve plateaus well before the iteration limit is reached. This premature convergence behavior is particularly problematic in construction scheduling, where small perturbations to the transition order can yield substantially different resource utilization patterns and completion times. By incorporating the SA acceptance

mechanism, a small proportion of individuals in the later stages of the search undergo mutation perturbations, and solutions that are slightly worse than the current best may still be accepted with a temperature-controlled probability. This prevents the population from collapsing into a single basin of attraction and allows the algorithm to continue exploring alternative scheduling sequences, ultimately leading to a shorter makespan than standard BSO achieves.

In recent years, BSO was employed in many fields [34]. Jordehi applied BSO to locate the optimal settings and locations of devices in electric power systems [35]. Chen used BSO to design the wireless sensor networks [36]. Hao employs a hybrid BSO to address the distributed CPO, and comparative analysis reveals its efficiency for the problem [37]. Bhatt used BSO to solve CPO in the cloud environment and compared the proposed model with GA and PSO to prove its better performance [38]. Hou proposed a multi-objective brainstorming optimization algorithm to solve the multi-objective integrated distributed flow shop scheduling problem [39]. As a population-based approach, BSO requires a careful balance between exploitation and exploration to function effectively [40]. Due to its focus on global searching, the benefit of exploration is that it increases the opportunities to gain global optimization. On the one hand, given that exploitation focuses on local searches, optimization offers rapid convergence. However, on the other hand, its drawback is a slow rate of convergence due to premature convergence and reduced opportunities to find the global optimum. These facts motivate us to investigate the hybridization of BSO algorithms with local search algorithms. Any local search will lead to fast convergence. Controlling the search progress in the optimization algorithm is crucial in order to avoid premature convergence and help the algorithm escape from the local optimum.

3. Methodology of Scheduling

3.1. Problem Characterization Based on Timed Petri Net

Optimizing the complex construction processes of large-scale projects is crucial for enhancing efficiency, reducing costs, and ensuring safety while meeting stringent deadlines and quality standards. Complex construction projects such as building construction comprise a combination of series events and materials. Complex construction projects require matching and optimizing different construction materials, construction processes, and procedures. This is essentially a resource allocation and scheduling problem. Specifically, this involves resource constraints and time constraints. First, ensure that the required construction materials are available at the right time and quantity to meet the needs of specific processes and procedures. Second, arrange the sequence and duration of the procedures reasonably to ensure that the project is completed within the scheduled time. Under the constraints of resources and time, minimize the total cost and improve construction efficiency.

When the processing procedures are certain, in actual production, different batches of raw materials have different processing times in the same processing procedures. As the number of batches increases, the production method of arranging batch numbers in sequence may result in a long production period and cause production to be overdue. Therefore, the workshop needs to optimize the production scheduling of the batches to be processed before carrying out production tasks and determine the production batches through appropriate decision-making methods with the minimum production period, as the goal is to shorten the production period. Such a problem can be considered as a job shop scheduling problem involving coordinating construction materials and construction technics. This process can be viewed as optimizing and scheduling multiple construction materials within defined time and equipment constraints. It is a combination of discrete processes with timestamps.

The overall methodology of the proposed approach is illustrated in Figure 1, which covers four main stages: timed Petri net modeling and verification, scheduling optimization formulation, improved BSO algorithm execution with SA local search, and output of the optimized scheduling plan. The following sections describe each stage in detail.

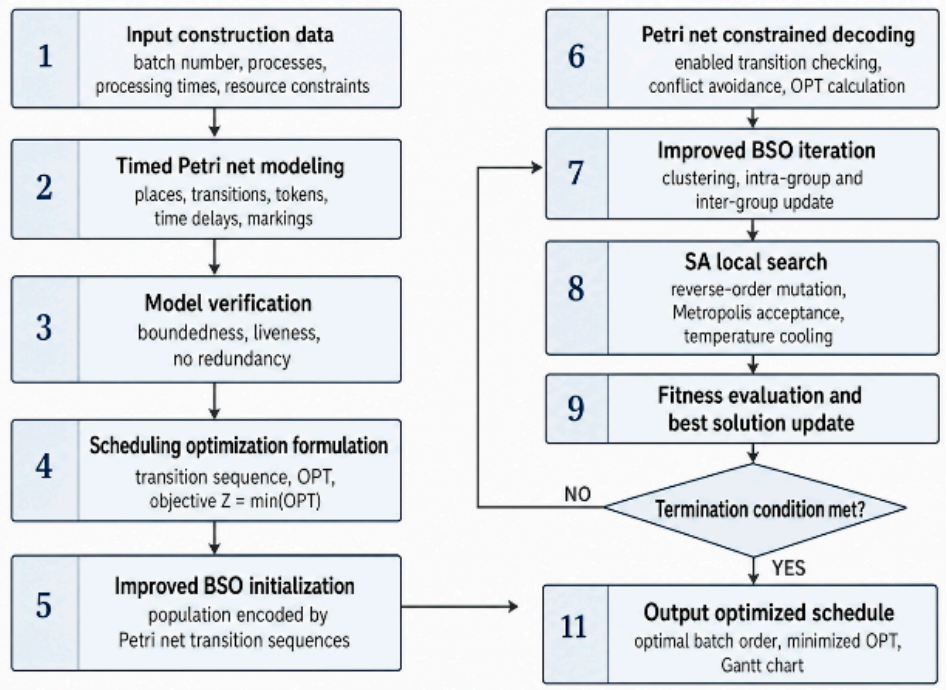


Figure 1. Overall flowchart of the proposed method.

Timed Petri net (TPN) is employed to represent both discrete and continuous processes. It categorizes places and transitions into continuous and discrete types [41]. Continuous elements describe the construction process, while discrete elements manage resource allocation. This hybrid approach, incorporating delayed and instantaneous transitions, effectively models time-dependent system performance. The elements contained in the TPN are defined in Equation (1). The definition of each symbol is shown in Table 1. Table 2 shows the elements in places and transitions.

$$K = (P, T, I, O, M_0, DT) \tag{1}$$

Table 1. Six-tuple interpretation.

Symbol	Meanings
P	The set of places.
T	The set of transitions.
I	I: $P \times T \longrightarrow N$ is the input function that defines the weight set of directed arcs from place P to transition T.
O	O: $T \times P \longrightarrow N$ is the output function that defines the weight set of directed arcs from transition T to place P.
M_0	Represents the initial identification of the place, that is, the distribution matrix of tokens in the place at the initial moment.
DT	The delay set of all places; the delay of the corresponding place can be zero or a positive real number.

Table 2. The elements in places and transitions.

Symbol	Meanings
P^r	Construction material set.
P	The term of construction technics set.
P^0	P^0 is raw material.
P^f	P^f is the final product set.
t^0	t^0 represents the beginning of processing activities.
t^f	t^f is the end of processing activities.

The construction process represented by the TPN actually refers to the evolution process of the attributes of a place token and the transformation between different transitions. In this paper, a token refers to the timestamp of resource allocation. The place refers to the different construction technics involved in the construction process. Taking building construction as an example, the token refers to the materials, such as rebar. The place refers to earthwork construction and other technics. Based on the above analysis and construction of the evolution structure, the mathematical definition of the TPN is obtained, which is expressed as follows:

Definition 1. *M_f represents the identification quantity of the place after the evolution of the Petri net, that is, the distribution matrix of tokens in the place at the end. D^+ and D^- are the pre- and post-correlation matrices of the Petri net structure, which are composed of the number of tokens in the corresponding place P when the transition T can occur. $D = D^+ - D^-$ represents the correlation matrix.*

Definition 2. *If the delay of a storage place is not 0, then the maximum identifier of the storage place is 1. For any transition T in the TPN model, it needs to satisfy the condition of Equation (2), and the residence time of the token in each input place in the place where it is located must be the time delay d_i of the place, then the transition can be inspired. T^* is the transition excitation sequence in T that satisfies the transition excitation conditions.*

Definition 3. *In the timed Petri net, each processing step in the production line is represented by a pair of transitions t^0 and t^f and a place P. The place P represents the processing operation. A single process completed in the equipment place R can be described by Figure 2, where P^0 and P^f represents raw materials and finished products, and the resource place R represents machines.*

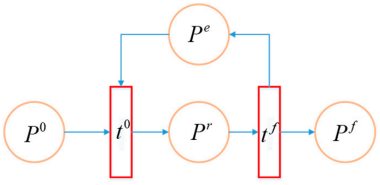


Figure 2. The model of a single process.

Definition 4. *In the Petri net, the element x_i in the vector $X(m)$ represents the moment when the current transition is triggered. The element v_i in the vector $V(m)$ represents the moment when the next transition is triggered, and its value is equal to the sum of the time delay of the place and the time when the token arrives at the place. During the evolution of the system, it is necessary to ensure that the processing machine remains idle when the next process is carried out after the current machine is completed. The $X(m)$ and $V(m)$ are defined as Equations (2) and (3).*

In Equation (2) and Equation (3), M_k denotes the current Petri net marking at step k, τ denotes the time delay of the corresponding place, and f_k and f_{k+1} denote the firing times of the current and next transitions, respectively.

$$v_i = \left\{ \begin{array}{l} \max(d_i + x_i), \ \ P_i \in O_p(t_i) \\ v_i \end{array} \right\} \tag{2}$$

$$x_{i+1} = \left\{ \begin{array}{l} v_i \ \ P_i \in O_p(t_i) \\ x_i \end{array} \right\} \tag{3}$$

$$OPT = \sum_{i=1}^{m} v_i \tag{4}$$

Definition 5. *Equation (2) shows that when the transition satisfies the excitation condition to enable the transition, the transition t_i can be excited and transfer the token to the place P_i. After the token reaches the place P_i, it must stay at least $d_i + x_i$ unit time before it can be used for excitation before triggering the next transition. Equation (3) is the updated method of the vector, indicating the moment when the next transition is triggered. In Equation (4), OPT represents the production time of the production line processing sequentially according to the transition excitation sequence.*

3.2. A Scheduling Model Based on Timed Petri Net

In the CPO problem, the raw materials will be organized into different batches and stored in specific areas according to corresponding models and production processes. In this paper, we assume that the warehouse can store enough raw materials and will not affect the construction progress. The scheduling problem based on multi-batch construction can be described as the construction plan of j batches. The CPO problem solved in this paper is to determine the construction sequence of each batch with the minimum OPT of the construction line processing period as the scheduling goal under the conditions of satisfying constraints. Here, the individual completion time of batch y is defined as $v_y = C(\sigma_y, J)$, and the total construction period is $OPT = \sum_{y=1}^{Y} v_y$. The optimization objective $Z = \min(OPT)$ thus minimizes the sum of all batch completion times. The optimization objectives of the construction scheduling mathematical model based on the proposed TPN can be established as Equations (5)–(11). Table 3 shows the meaning of the formula.

$$\sigma_y = \left\{ t_{y1}, t_{y2} t_{y3}, \ldots\ldots t_{yk} \right\} \ y = 1, 2, 3 \ldots\ldots Y \ k = 1, 2, 3 \ldots\ldots m \tag{5}$$

$$OPT = \sum_{y} \sum_{j} C(\delta_y, j) = \sum_{i}^{m} v_i \tag{6}$$

$$Z = \min(OPT) = \min(\sum_{i=1}^{m} v_i) \tag{7}$$

$$C(\delta_y, j) = B(\delta_y, j) + Q(\delta_y, j) \tag{8}$$

$$C(\delta_y, j) \leq B(\delta_y, j+1) \tag{9}$$

$$C(\delta_y, j) - C(\delta_e, j) \geq Q(\delta_y, j) \tag{10}$$

$$C(\delta_y, j) - B(\delta_y, j) \geq Q(\delta_y, j) \tag{11}$$

Table 3. Symbols of optimization module.

Symbol	Meaning
Y	Batch number
J	Number of processes
k	Number of transitions
σ_y	Batch sequence of the construction scheduling problem
t_{yk}	Set of change set at processing the y batch
$C(\delta_y, j)$	The completion time of batch σ_y
$B(\delta_j, k)$	The start time of batch y for construction technic j
$Q(\delta_j, k)$	The processing time of batch y for construction technic j
OPT	Total construction period, $OPT = \sum_{y=1}^{Y} v_y$
Z	Optimization objective, $Z = \min(OPT)$
e	Index of another batch in Equation (10), where $e \neq y$, indicating that batches y and e cannot occupy the same equipment simultaneously

Equation (8) means that if the same batch of workpieces currently has a processing process, the completion time of the process is equal to the sum of the start time of the process and the processing time, otherwise the completion time is equal to the start time. Equation (9) means that for the previous process of the same workpiece, the relationship between the completion time of a process and the start time of the next process means that you need to wait for the completion of the previous process before processing the next process. Equation (10) expresses that the processing equipment in any processing process can only process one batch of workpieces at a time, that is, any two different batches of workpieces y and e cannot be processed in the same process at the same time. Equation (11) is expressed as the completion time of any process minus its start time and cannot be less than its processing time.

4. Scheduling Optimization Based on BSO Algorithm

4.1. Principle of BSO

When a single processing batch with the number of process steps is processed on the Petri net, its transitions are triggered sequentially. However, when the amount of scheduling tasks increases, the transitions of different batches will produce different scheduling results and transition competition will occur. In order to achieve the production scheduling goal and obtain the optimal scheduling result, the brainstorm optimization algorithm (BSO) is introduced into the Petri net, and the corresponding constraint processing mechanism is designed in the encoding and decoding. Avoid transition competition and use the simulated annealing local search mechanism in the later stage to achieve the scheduling goal of minimizing the batch completion time, thereby improving the optimization performance of the BSO algorithm.

This paper adopts the method of combining models with algorithms, using model elements to construct the optimization objective function of the algorithm to solve the scheduling strategy. The pseudocode of the production scheduling algorithm is shown in Algorithm 1. BSO algorithm pseudocode flow.

Algorithm 1. Initial BSO algorithm

Input: population size, population (pop), M_0 M_f, D, DT
Output: OPT, transition excitation Sequence T^*
1. ***initialization***: Randomly generate individuals of population size composed of $T = \left(t_{y1}, t_{y2,} \ldots\ldots, t_{yk}\right)$
2. ***for*** iteration = 1 to max number of iterations
3. ***for*** i = 1 to population size
4. $OPT(i) = f_{opt}(T, M_0, M_f, D, DT)$
5. ***End for***
6. $[cluster, center] = f_{k_means}(pop, OPT)$
7. ***if*** $rand < P_a$ (Cluster center update probability)
8. *center = center_new*
9. ***end if***
10. ***if*** $rand < P_b$ (Selection probabilities were discussed between internal groups) Generates new individuals new_T
11. $OPT(new_T) = f_{opt}(new_T, M_0, M_f, D, DT)$
12. ***else*** Intergroup perturbations generate a new individual new_T
13. $OPT(new_T) = f_{opt}(new_T, M_0, M_f, D, DT)$
14. **if** iteration > $\lfloor 0.05 \cdot N_{iter} \rfloor$, execute SA local search
15. end if
16. ***if*** $OPT(new_T) < OPT(T)$
17. $T = new_T$
18. end if
19. end for

4.2. Encoding and Decoding of BSO

Considering that as the construction tasks increase, different batches of transitions may be triggered, leading to transition conflicts, it is essential to integrate a Petri net and process the encoded solution set. In the encoding step, a corresponding constraint handling mechanism is developed to avoid transition conflicts. Precisely, in the population initialization phase, a method is employed to randomly shuffle a finite number of transition entities $T = \left(t_{y1}, t_{y2,} \ldots\ldots, t_{yk}\right)$ from the Petri net and combine them into different individuals, which serves as the encoding method to create the initial population $population = \{T_1, T_2, \ldots\ldots, T_i\}$.

During the individual decoding process, a set of transitions T^* that can occur is determined based on the triggering conditions of transitions and resource constraints. The individual decoding algorithm flow is shown in Algorithm 2. Transition sequence decoding.

In the proposed BSO combining the Petri net, we use two individual update methods: intra-group discussion and inter-group discussion. For the intra-group update, a reverse-order subsequence mutation is applied. Specifically, two positions i and j (i < j) are randomly selected in the current individual. The subsequence of transitions located between positions i and j is then reversed in place, while all transitions outside this range remain unchanged. For example, if positions 4 and 10 are selected, the transitions at positions 4 through 10 are reordered in reverse, producing a new individual that differs locally from the original while retaining all transitions without duplication or omission. The principle is illustrated in Figure 3.

Algorithm 2. Individual decoding computes the fitness values

Step1: Determine initial values M_0 M_f dT D^- D^+
Step2: Calculate the association matrix $D = D^+ - D^-$;
Step3: ***if*** $M_k \neq M_f$, proceed to Step 4; otherwise, go to Step 9.
Step4: Evaluate the set of transitions that can be triggered under the marking $M_k \geq D^-(:, t_{yk})$
Step5: Determine the batch σ_y of the current stimulated change t_{yk} and the processing process;
Step6: If the current t_{yk} batch is σ_y and the processing process is j, the change weight of the processing batch on process j except σ_y needs to be updated to constraint$(1, t_{ek}) = 0$ Update the transition weights of transitions in $j + 1$ to constraint$(1, t_{y(k+1)}) = 0$; The change weight of the batch is σ_y and processed on process j + 1 is updated to constraint$(1, t_{y(k+1)}) = 1$
Step7: Record one triggered transition v_i at a time and calculate the time x_{i+1} at which the next transition will be triggered.
Step8: When a batch processing is completed, calculate its completion time $C(\delta_y, j)$, update the marking m_{k+1}, and return to Step 3.
Step9: Update the method as above to calculate the $OPT = \sum_y \sum_j C(\delta_y, j)$ when all batches are processed

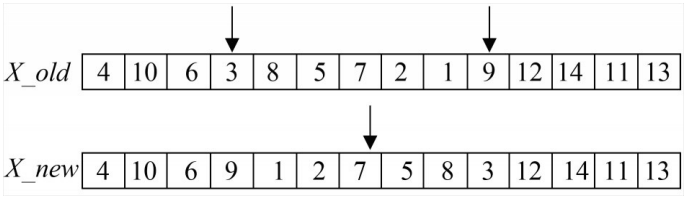


Figure 3. Update method of individuals in the group.

When using the merge operator to generate new individuals between groups, two parent individuals are combined using a position-based crossover. Specifically, a set of positions is randomly selected from individual 1, and the transitions at those positions are copied directly into the same positions of the new individual. The remaining positions of the new individual are then filled, from left to right, with the transitions taken from individual 2 in their original order but excluding any transitions already copied from individual 1. This procedure, as shown in Figure 4, ensures that each transition appears exactly once in the new individual, preserving the feasibility of the encoded sequence.

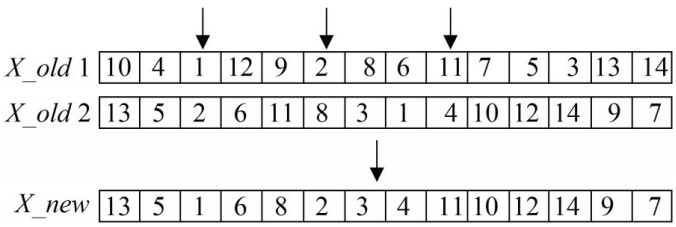


Figure 4. Update method of individuals between groups.

4.3. The Improved Local Search Mechanism

In the standard BSO algorithm, because the way to select individuals is to randomly select according to probability parameters, the population diversity can be maintained at a reasonable level in the early stage. With the continuous update of individuals, the cluster center of each subgroup is basically stable, and individuals in the adjacent area may be

excluded so that the optimal solution of the current iteration no longer changes, resulting in a premature algorithm.

In this paper, we introduced the SA algorithm into optimization calculation. The core of the SA algorithm is to determine whether to accept new individuals with poor target values after mutation based on the Metropolis probability acceptance criterion, that is, the "accepting a degraded solution with a certain probability" mechanism. Therefore, it has good local search ability and is easy to jump out of the local optimal point when applied in optimization calculation [42]. Therefore, this paper adds the local search mechanism of the SA algorithm to the BSO algorithm and searches for the global optimality of the objective function in the solution space by performing mutation perturbations on a few individuals in the later stage of the algorithm and combining the probability jump method. The Metropolis probability value is calculated in Equation (12), where ΔOPT denotes the difference in objective values between the newly generated individual and the current one, Tem denotes the current temperature, and α is the temperature control coefficient with a value typically ranging from 0.5 to 0.99. In Equation (13), Tem(k) denotes the temperature at the k-th cooling step, and α is the same cooling coefficient. The algorithm flow of the Metropolis probability acceptance criterion is shown in Algorithm 3. Metropolis probability acceptance criterion.

Algorithm 3. The algorithm flow

Step 1: ***Randomly*** perturb $n_p = \max(1, \lfloor 0.1 \cdot \mid pop \mid \rfloor)$ individuals in pop using the reverse-order frameshift mutation, and calculate the objective function value OPT;
Step 2: ***If*** OPT (new) < OPT (old), directly replace the old individual; ***If*** OPT (new) > OPT (old), calculate the Metropolis probability value according to Formula (12) and generate a random number random (0,1); ***If*** random (0, 1) < Metropolis, accept the degraded individual, that is, the individual with a larger OPT value, and go to step 4;
Step 3: ***If*** OPT (new) < OPT (old), directly update the individual and go to step 4;
Step 4: ***Update*** the temperature parameter according to Formula (13).

$$Metropolis = \left\{ \begin{array}{l} 1, OPT(new) \le OPT(old) \\ \exp(-\frac{OPT(new)-OPT(old)}{Tem(i)}), OPT(new) > OPT(old) \end{array} \right\} \quad (12)$$

$$Tem(i+1) = \boldsymbol{\alpha} Tem(i) \quad (13)$$

Several implementation details are provided here to ensure reproducibility. (1) SA activation: the local search in Algorithm 3 is triggered only after the iteration count exceeds $\lfloor 0.05 \cdot N_{iter} \rfloor$, i.e., in the latter half of the run. (2) Infeasible repair: no dedicated repair operator is required because the decoding procedure (Algorithm 2, Steps 3–6) guarantees feasibility by construction—at each decoding step, only transitions enabled under the current Petri net marking $M_k \ge D^-\left(:, t_{yk}\right)$ are candidates, so the decoded sequence T^* inherently satisfies all precedence and resource constraints. (3) Initial temperature and cooling: the initial temperature is set as $Tem(1) = -\overline{\Delta}/\ln P_0$, where $\overline{\Delta}$ is the mean absolute OPT difference between randomly paired individuals in the initial population and $P_0 = 0.9$; the cooling coefficient defaults to $\alpha = 0.95$.

To briefly illustrate the encoding–decoding procedure, consider Y = 2 batches, J = 3 processes, and equal processing times $Q(\sigma_y, j) = 2$ for all y, j. The six transitions $\{t_{1,1}, t_{1,2}, t_{1,3}, t_{2,1}, t_{2,2}, t_{2,3}\}$ are randomly shuffled into an individual $T = (t_{1,1}, t_{2,1}, t_{1,2}, t_{2,2}, t_{1,3}, t_{2,3})$. Decoding follows Algorithm 2: both $t_{1,1}$ and $t_{2,1}$ are initially enabled; $t_{1,1}$ fires at time 0 giving $C(\sigma_1, 1) = 2$; $t_{2,1}$ fires next at time 2 giving $C(\sigma_2, 1) = 4$; continuing in this way yields $C(\sigma_1, 3) = 6$ and $C(\sigma_2, 3) = 8$, so OPT = 6 + 8 = 14. For the

intra-group update, reversing positions 2–4 of T gives $T' = (t_{1,1}, t_{2,2}, t_{1,2}, t_{2,1}, t_{1,3}, t_{2,3})$, which is decoded and evaluated identically. The inter-group and SA perturbations follow the same decoding pipeline.

5. Case Study

5.1. Parameter Settings

This paper uses a real building construction process as an example to illustrate the effectiveness of the proposed method as shown in Figure 5. The example is from Beijing Urban Sub-Center Comprehensive Transportation Hub, a critical high-speed railway transit hub and the largest transportation hub in Asia. This project was selected for three reasons. First, its seven-process workflow and 20 construction batches provide sufficient complexity to stress-test the proposed algorithm. Second, the strict equipment-sharing and precedence constraints of this project closely match the structural assumptions of the proposed TPN model. Third, the actual construction records of this project were accessible to the authors, allowing the use of real processing time data rather than synthetically generated instances.

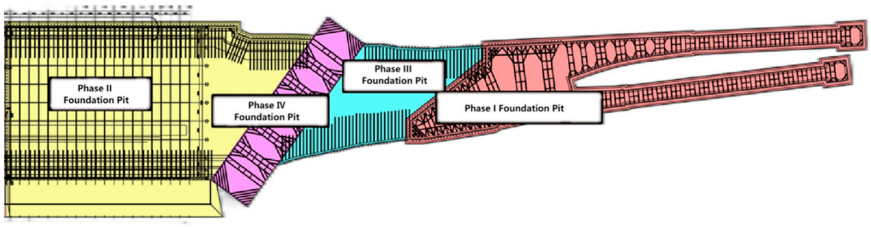


Figure 5. The layout of the example.

The seven processes follow the following sequence, as shown in Figure 6: reinforced concrete foundation → earthwork → base plate construction → sidewall construction → roof construction → electromechanical installation → secondary decoration. It should be noted that the reinforced concrete foundation here refers to the construction of retaining piles and supporting structures from the ground surface before excavation begins, which is a standard top-down construction practice for deep urban foundation pits where the perimeter retaining structure must be in place first to ensure slope stability and protect adjacent facilities during subsequent earthwork excavation.

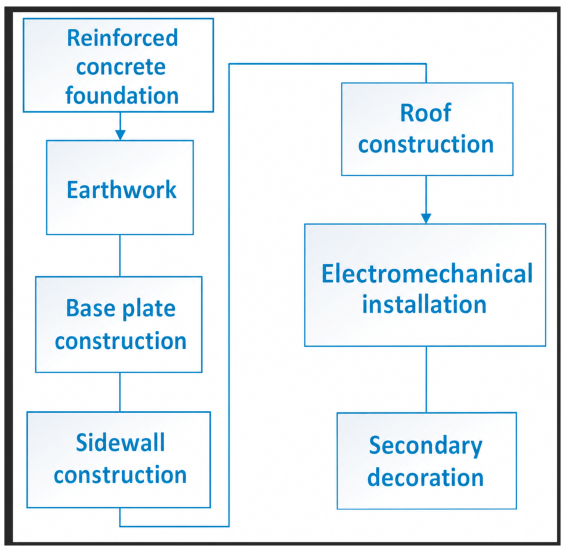


Figure 6. Construction sequence of the construction process.

This paper uses the sequence in Figure 6 as a prototype to establish a general scheduling Petri net simulation model for each batch participating in processing and production, as shown in Figure 7. In this model, raw materials are fed from place P1, undergo seven processing steps, and then finished materials are output from P15. The meanings of places are shown in Table 4. P_{16}–P_{22} represent the seven equipment resource places, one assigned to each of the seven construction processes. t_i represents the allocation of corresponding equipment when i is odd, and it represents the unloading of equipment and entering the next process when i is even.

Table 4. Interpretation of places and transitions in the example.

Symbol	Term	Symbol	Term
p_1	Loading and process begins	p_8	Sidewall construction processing
p_2	The reinforced concrete foundation	p_9	Process completed and enter the next step
p_3	Process completed and enter the next step	p_{10}	Roof construction
p_4	The earthwork begins	p_{11}	Process completed and enter the next step
p_5	Process completed and enter the next step	p_{12}	Electromechanical installation
p_6	Quenched processing base plate construction	p_{13}	Process completed and enter the next step
p_7	Process completed and enter the next step	p_{14}	Secondary decoration processing

The processing time data in Table 5 are derived from the actual construction records of the project, where each value represents the scheduled working hours for the corresponding batch and process. Due to confidentiality requirements, minor adjustments were applied to some entries while preserving the overall statistical characteristics of the original data. All values are in working hours (h).

Table 5. Production task data.

Batch	Technic 1	Technic 2	Technic 3	Technic 4	Technic 5	Technic 6	Technic 7
01	110	174	121	183	102	190	150
02	80	120	176	174	199	160	115
03	86	167	148	160	166	138	130
04	95	97	76	64	110	181	173
05	102	87	86	80	90	83	75
06	113	80	42	99	70	90	93
07	85	69	72	54	80	90	160
08	84	69	120	24	96	89	62
09	66	50	88	46	63	76	150
10	74	97	74	82	53	61	95
11	100	170	120	103	102	100	150
12	99	120	176	174	199	160	115
13	89	170	158	120	160	140	130
14	102	97	76	64	110	181	173
15	111	87	86	80	90	83	75
16	77	80	42	99	70	90	93
17	86	169	87	65	80	92	130
18	75	79	120	54	106	88	100
19	94	50	88	46	63	76	150
20	78	97	74	82	53	61	95

There is a total of 20 batches of production tasks. The processing time of each process in each batch of operations on the production line is given for different batches.

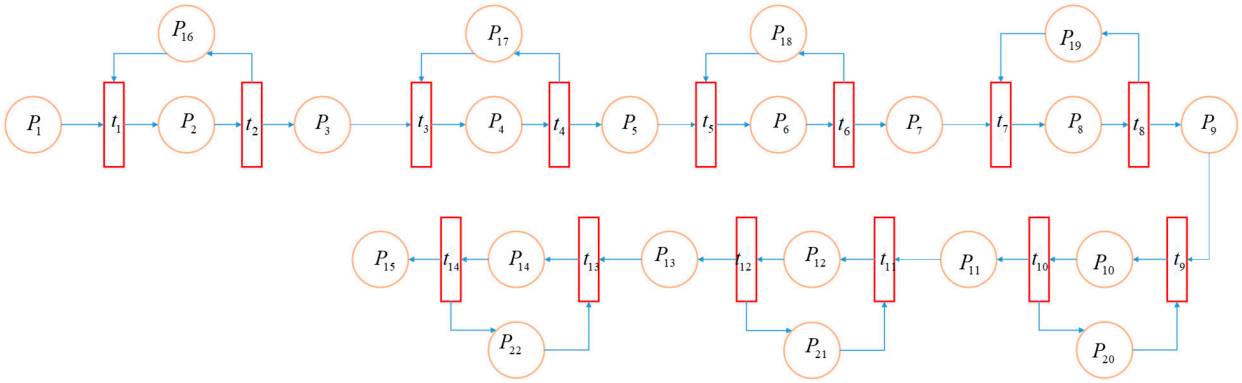


Figure 7. Timed Petri net of construction process.

5.2. Analysis of Timed Petri Net

The analysis of the rationality of the timed Petri net mainly includes reachability identification diagrams and correlation matrices. Among them, the accessibility identification diagram is based on graph theory and pays more attention to the logical rationality of the model. The logical structure of the Petri net is mapped and established according to the processing sequence of production factors involved in resource allocation in the actual production line. This simple corresponding relationship can better assist the analysis of JSPs.

Therefore, this paper conducts a rational verification on the timed Petri net based on the analysis method of the reachable identification diagram. The reachable identification graph of the Petri net refers to a directed graph composed of the reachable identification set as the vertex set and the direct reachability relationship between the identifications as the arc set. In the reachable identification graph, each vertex is the identification vector of the corresponding state, and the transition that causes the identification change is marked on one side of the arc. Figure 8 presents the reachable marking graph in a simplified form. In Figure 8, the initial marking is $M_0 = (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1)$, and the final marking is $M_f = (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1)$. The intermediate states $M_1, M_2, \ldots, M_{13}$ correspond to the reachable markings generated by sequential transition firing.

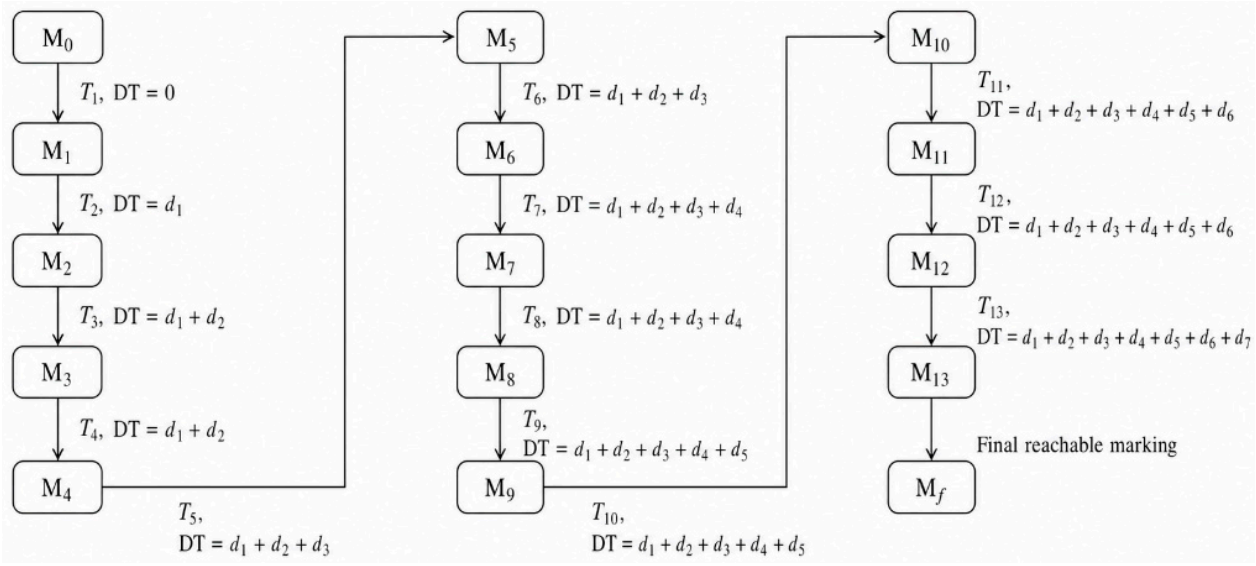


Figure 8. Reachable marking graph of timed Petri net.

The initial state of the Petri net in Figure 8 is identified as M_0 = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1). The end state identifier M_f = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1) is obtained after the driver evolution of the delay and transition ends. From Figure 8, we can get the main characteristics of the proposed Petri net for the JSP problem:

(1) The Petri net model is bounded. For all places except the machine resource places, each place holds at most one token at any time. A token appearing in the terminal place indicates the completion of all processing steps, and no node in the model is executed more than once, confirming the boundedness of the net.
(2) The Petri net model is active and is identified by the reachable graph. It can be seen that changes under any reachable identifier have the right to occur, so the entire model system is also active.
(3) There is no redundancy in the Petri net model. All variables involved in Figure 7 appear in Figure 8, which shows that all changes in the model are necessary. Importantly, there are no nodes that will not be executed, references to instances, or complete nodes that do not contribute.

To sum up, the Petri net model established in Figure 7 is a correct and reasonable network model, so the scheduling problem can be transformed into obtaining the Petri net network model's transition-induced sequence problem.

Several practical constraints specific to the railway hub construction project are reflected in the proposed model. First, equipment sharing among different construction batches is explicitly captured by the resource places P_{16}–P_{22} in the Petri net, which prevents concurrent allocation of the same machinery to multiple batches. Second, the strict precedence among the seven construction processes (e.g., earthwork must precede base plate construction) is enforced through the token-flow mechanism, ensuring that no process begins before its predecessor is completed. Third, the model assumes single-shift daily operation, which is consistent with the safety regulations of the actual project site.

From a managerial perspective, the optimized batch sequence suggests that shorter early-stage batches should be prioritized to release shared equipment earlier, thereby reducing idle waiting time in subsequent processes. The 531-working-hour reduction corresponds to approximately 22 calendar days under the project's five-day work week, which has direct implications for contract compliance and cost control. The proposed approach can also serve as a decision-support tool for site managers during the preconstruction planning phase, providing a near-optimal scheduling plan within a practical computation time.

From a policy perspective, large-scale transportation infrastructure projects such as high-speed railway hubs are typically funded by government agencies and subject to strict delivery deadlines and budget constraints. The results of this study suggest that adopting optimization-based scheduling methods at the project planning stage can meaningfully reduce construction periods and associated costs. Relevant authorities and procurement agencies may therefore consider mandating or incentivizing the use of such tools in the tendering and execution phases of major infrastructure contracts. In addition, the reduction in construction duration also implies a shorter period of site disruption to surrounding communities and transport networks, which is a consideration of direct public interest in urban infrastructure projects.

5.3. Results and Comparison

The initial individual composed of transition sequences generates transition excitation sequences under the excitation conditions and resource scheduling constraints. When the state marking evolves to the end marking, the scheduling ends, and the function value required according to the scheduling order of the current transition excitation sequence is

calculated. Through the continuous iteration and optimization strategy of the algorithm, the optimal individual and the corresponding scheduling batch are finally obtained.

Under the initial marking M0 = (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1), the place P1 has a token marking that there is a construction material; in the construction sites P16, P17, P18, P19, P20, P21, and P22, there is a token, indicating that the required equipment resources can be scheduled; under the end marking Mf, the P15 should have a token to indicate the completion of construction. The scheduling algorithm parameter settings used in this paper are shown in Table 6. The parameter settings of the algorithm refer to [43].

Table 6. Parameters of BSO.

Main Parameter	Value
pop	200
$Maxit$	500
m	2
$p_{clustering}$	0.6
Tem	100

During the experiment, the production task data table is tested in two ways. One is the batch scheduling sequence given in the actual production; the total construction period is 3433. As shown in Figure 9, different colors are used to represent different batches to draw the scheduling Gantt chart of each batch, in which the vertical axis represents the processing process. The horizontal axis represents the scheduling period of the production task. Taking batch J1 as an example, the batch has a total of 1030, meeting the task data table in each process processing completion period. When the first process of batch J1 is finished, machine M1 is converted from the processing state to the idle state. At this time, the batch is loaded and the first process of J2 is processed by M1, and so on, to complete the processing tasks of all batches.

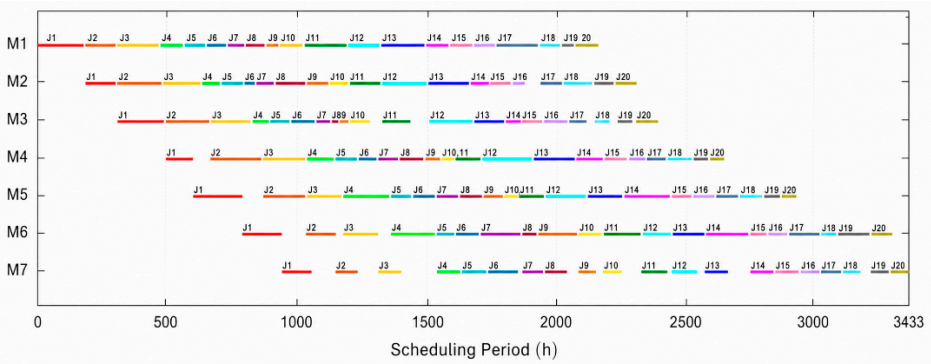


Figure 9. Gantt chart before optimization.

The time-based BSO algorithm proposed in this paper is used to optimize the scheduling of production tasks. The scheduling results show that the total construction period is 2902 working hours, as shown in the Gantt chart of Figure 10, and the batch scheduling order is 7,6,9,14,19,4,12,20,18,1,2,11,16,15,13,3,8,5,17,10; the transition excitation sequence of each production batch t_{ji} is t (y, 1), t (y, 2), t (y, 3), t (y, 4), t (y, 5), t (y, 6), t (y, 7), t (y, 8), t (y, 9), t (y, 10), t (y, 11), t (y, 12), t (y, 13), t (y, 14), where y is the batch number.

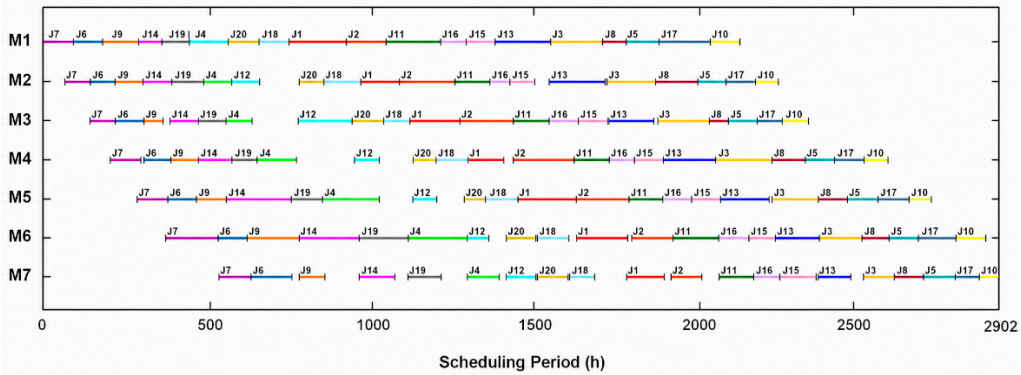


Figure 10. Gantt chart after optimization.

As shown in Figure 11, both algorithms start from the same initial population. The BSO+SA algorithm converges rapidly, reaching a near-optimal solution within approximately 35 iterations and attaining its final value of 2902 working hours at around iteration 150. In contrast, the standard BSO improves more slowly and stagnates at 3021 working hours. This comparison demonstrates that the embedded SA mechanism not only accelerates convergence but also enables the algorithm to escape local optima and achieve a better final solution.

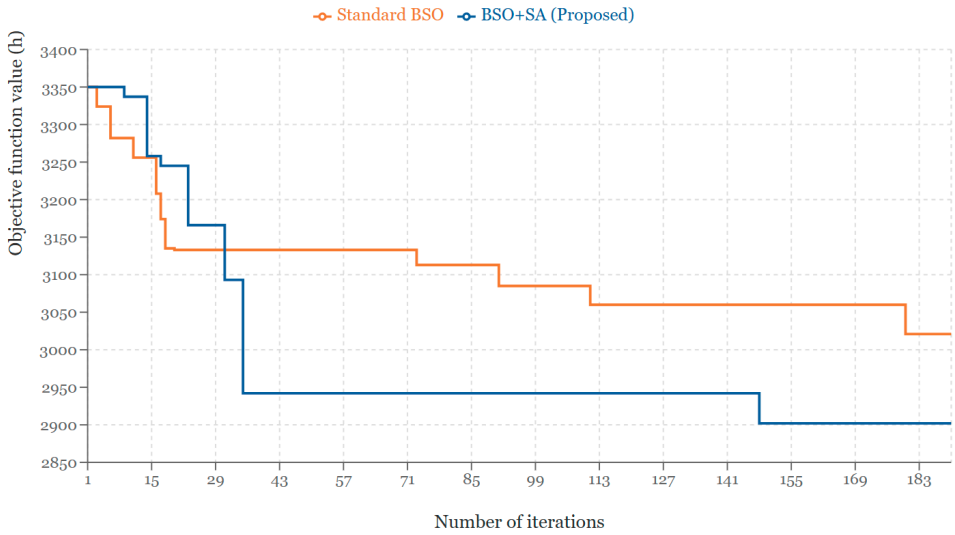


Figure 11. The behavior of the BSO algorithm and the improved BSO.

To evaluate the contribution of each component and benchmark the proposed method, five algorithms were compared on the 20-batch instance: BSO+SA, standard BSO, GA, PSO, and SA. All algorithms shared the same TPN-based encoding and decoding framework, and each was executed independently for 30 runs. The best OPT, mean OPT, standard deviation, worst OPT, and average CPU time are reported in Table 7. BSO+SA achieves the lowest best and mean OPTs of 2902 and 2941, with a standard deviation of 25.1, outperforming all competitors. The reduction in both mean OPT and standard deviation relative to standard BSO confirms that the SA local search effectively suppresses premature convergence without introducing prohibitive computational overhead.

Table 7. Comparison results of different algorithms.

Algorithm	Best	Mean	Std	Worst	Avg. CPU(s)
BSO+SA (Proposed)	2902	2941	25.1	3008	46.3
Standard BSO	3021	3089	50.3	3201	39.1
PSO	3048	3172	62.4	3339	38.6
GA	3065	3215	76.8	3421	41.7
SA	2987	3196	88.7	3459	53.4

To assess scalability, experiments were further conducted on a small instance (10, constructed from the first ten rows of Table 5) and a large instance (30, generated by proportional perturbation of the original data), with results reported in Table 8. BSO+SA consistently achieved the lowest OPT across all three scales, and its relative improvement over the sequential schedule increased from 15.0% at Y = 10 to 17.8% at Y = 30. The performance gap between BSO+SA and the next best competitor also widened with problem size, indicating that the proposed method scales well and its advantage becomes more pronounced as the solution space grows.

Table 8. Scalability experimental results.

Algorithm	Y = 10, J = 7	Y = 20, J = 7	Y = 30, J = 7
Sequential	1789	3433	5214
BSO+SA	1521	2902	4287
Standard BSO	1594	3021	4536
PSO	1617	3048	4587
GA	1632	3065	4621
SA	1558	2987	4378

6. Conclusions

This paper proposed a timed-Petri-net-based scheduling optimization framework for large-scale construction projects, with the goal of minimizing the total construction period. Three main findings are drawn from this study. First, the timed Petri net model was shown to be a valid and effective tool for representing the multi-batch, multi-process construction workflow of the Beijing Urban Sub-Center Comprehensive Transportation Hub, passing boundedness, liveness, and redundancy checks through reachability analysis. Second, the problem-specific encoding and decoding method designed for Petri net transition sequences guarantees solution feasibility by construction, eliminating the need for repair procedures that are commonly required in GA- and PSO-based approaches. Third, the proposed BSO+SA algorithm, which embeds a simulated annealing acceptance mechanism into the latter half of the BSO search process, outperformed standard BSO, GA, PSO, and standalone SA across all tested problem scales. On the 20-batch case study, it reduced the total construction period from 3433 to 2902 working hours, representing a 15.47% improvement over sequential scheduling. The performance advantage of BSO+SA over competing methods widened as problem size increased from Y = 10 to Y = 30, indicating good scalability.

This study has several limitations that should be acknowledged. The current model assumes single-shift daily operation and does not account for resource disruptions such as equipment breakdowns, labor shortages, or weather-related delays that frequently occur in real construction environments. The model also assumes that processing times are deterministic and known in advance, whereas in practice they are subject to uncertainty. In addition, the optimization objective in this paper is limited to minimizing the total

construction period without simultaneously considering cost, resource leveling, or carbon emissions. Finally, the case study is based on a single project, and the generalizability of the results to other types of construction projects has not yet been verified.

Several directions for future research are suggested. First, stochastic or robust versions of the proposed TPN model could be developed to handle uncertainty in processing times and resource availability, making the framework more applicable to real-world dynamic environments. Second, the optimization model could be extended to a multi-objective formulation that simultaneously minimizes construction duration, total cost, and resource fluctuation, using multi-objective variants of BSO or other evolutionary algorithms. Third, the proposed framework could be tested on other types of infrastructure projects, such as bridge construction or underground utility installation, to assess its broader applicability. Fourth, integrating the scheduling tool with building information modeling systems could provide a practical pathway for real-time rescheduling support on active construction sites.

Author Contributions: Conceptualization, W.W.; methodology, W.W.; Validation, W.W.; writing—original draft preparation, W.W.; supervision, E.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data used in this study are derived from an ongoing construction project and cannot be publicly released due to confidentiality requirements. Readers who wish to access the data may contact the corresponding author by email.

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

References

1. Fu, Y.; Hou, Y.; Wang, Z.; Wu, X.; Gao, K.; Wang, L. Distributed scheduling problems in intelligent manufacturing systems. *Tsinghua Sci. Technol.* **2021**, *26*, 625–645. [CrossRef]
2. Lohmer, J.; Lasch, R. Production planning and scheduling in multi-factory production networks: A systematic literature review. *Int. J. Prod. Res.* **2021**, *59*, 2028–2054.
3. Zhang, S.; Li, X.; Zhang, B.; Wang, S. Multi-objective optimisation in flexible assembly job shop scheduling using a distributed ant colony system. *Eur. J. Oper. Res.* **2020**, *283*, 441–460. [CrossRef]
4. Graves, S.C. A review of production scheduling. *Oper. Res.* **1981**, *29*, 646–675. [CrossRef]
5. Herrmann, J.W. *Handbook of Production Scheduling*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2006; Volume 89.
6. Parente, M.; Figueira, G.; Amorim, P.; Marques, A. Production scheduling in the context of Industry 4.0: Review and trends. *Int. J. Prod. Res.* **2020**, *58*, 5401–5431. [CrossRef]
7. Pan, L.; Yang, B.; Jiang, U.; Zhou, M. A Time Petri Net with Relaxed Mixed Semantics for Schedulability Analysis of Flexible Manufacturing Systems. *IEEE Access* **2020**, *8*, 46480–46492. [CrossRef]
8. Xu, S.Z. A PETRI NET-BASED HYBRID HEURISTIC SCHEDULING ALGORITHM FOR FLEXIBLE MANUFACTURING SYSTEM. *Int. J. Simul. Model.* **2019**, *18*, 325–334. [CrossRef]
9. Schild, A.; Rose, A.; Grotjahn, M.; Luck, B. A modular framework for optimal scheduling of industrial processes. *At-Automatisierungstechnik* **2022**, *70*, 31–37. [CrossRef]
10. Zhou, J.; Luo, J.; Lefebvre, D.; Li, Z. Modeling and Scheduling Methods for Batch Production Systems Based on Petri Nets and Heuristic Search. *IEEE Access* **2020**, *8*, 163458–163471. [CrossRef]
11. Huang, B.; Zhou, M.; Abusorrah, A.; Sedraoui, K. Scheduling Robotic Cellular Manufacturing Systems with Timed Petri Net, A* Search, and Admissible Heuristic Function. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 243–250. [CrossRef]
12. Liu, H.R.; Jiang, Z.B.; Fung, R.Y.K. Performance modeling, real-time dispatching and simulation of wafer fabrication systems using timed extended object-oriented Petri nets. *Comput. Ind. Eng.* **2009**, *56*, 121–137. [CrossRef]
13. Luo, J.L.; Wan, Y.X.; Wu, W.M.; Li, Z.W. Optimal Petri-Net Controller for Avoiding Collisions in a Class of Automated Guided Vehicle Systems. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 4526–4537. [CrossRef]
14. Zhang, P.; Song, S.; Niu, S.; Zhang, R. A Hybrid Artificial Immune-Simulated Annealing Algorithm for Multiroute Job Shop Scheduling Problem with Continuous Limited Output Buffers. *IEEE Trans. Cybern.* **2022**, *52*, 12112–12125. [CrossRef] [PubMed]

15. Ojstersek, R.; Brezocnik, M.; Buchmeister, B. Multi-objective optimization of production scheduling with evolutionary computation: A review. *Int. J. Ind. Eng. Comput.* **2020**, *11*, 359–376. [CrossRef]
16. Grosch, B.; Kohne, T.; Weigold, M. Multi-objective hybrid genetic algorithm for energy adaptive production scheduling in job shops. *Procedia CIRP* **2021**, *98*, 294–299. [CrossRef]
17. Babor, M.; Senge, J.; Rosell, C.M.; Rodrigo, D.; Hitzmann, B. Optimization of no-wait flowshop scheduling problem in bakery production with modified pso, neh and sa. *Processes* **2021**, *9*, 2044. [CrossRef]
18. Yağmur, E.; Kesen, S.E. Multi-trip heterogeneous vehicle routing problem coordinated with production scheduling: Memetic algorithm and simulated annealing approaches. *Comput. Ind. Eng.* **2021**, *161*, 107649. [CrossRef]
19. Anuar, N.I.; Fauadi, M.H.F.M.; Saptari, A. Performance Evaluation of Continuous and Discrete Particle Swarm Optimization in Job-Shop Scheduling Problems. In Proceedings of the International Conference on Recent Advances in Industrial Engineering and Manufacturing (ICRAIEM), Penang, Malaysia, 12–13 December 2018.
20. Dao, S.D.; Abhary, K.; Marian, R.; Goh, M. Optimisation of production scheduling for multi-product orders in VCIM systems using GA. *Soft Comput.* **2019**, *23*, 10199–10224.
21. Lim, K.C.W.; Wong, L.-P.; Chin, J.F. Simulated-annealing-based hyper-heuristic for flexible job-shop scheduling. *Eng. Optim.* **2022**, *55*, 1635–1651. [CrossRef]
22. Li, Y.; Wang, C.; Gao, L.; Song, Y.; Li, X. An improved simulated annealing algorithm based on residual network for permutation flow shop scheduling. *Complex Intell. Syst.* **2021**, *7*, 1173–1183. [CrossRef]
23. Mohammadi, M.; Musa, S.N.; Bahreininejad, A. Optimization of mixed integer nonlinear economic lot scheduling problem with multiple setups and shelf life using metaheuristic algorithms. *Adv. Eng. Softw.* **2014**, *78*, 41–51. [CrossRef]
24. Liu, L.-L.; Hu, R.-S.; Hu, X.-P.; Zhao, G.-P.; Wang, S. A hybrid PSO-GA algorithm for job shop scheduling in machine tool production. *Int. J. Prod. Res.* **2015**, *53*, 5755–5781. [CrossRef]
25. Mumtaz, J.; Guan, Z.; Yue, L.; Wang, Z.; Ullah, S.; Rauf, M. Multi-Level Planning and Scheduling for Parallel PCB Assembly Lines Using Hybrid Spider Monkey Optimization Approach. *IEEE Access* **2019**, *7*, 18685–18700. [CrossRef]
26. Mumtaz, J.; Guan, Z.; Yue, L.; Zhang, L.; He, C. Hybrid spider monkey optimisation algorithm for multi-level planning and scheduling problems of assembly lines. *Int. J. Prod. Res.* **2020**, *58*, 6252–6267. [CrossRef]
27. Tang, H.; Chen, R.; Li, Y.; Peng, Z.; Guo, S.; Du, Y. Flexible job-shop scheduling with tolerated time interval and limited starting time interval based on hybrid discrete PSO-SA: An application from a casting workshop. *Appl. Soft Comput.* **2019**, *78*, 176–194. [CrossRef]
28. Saophan, P.; Pannakkong, W. A Genetic Algorithm Based Artificial Neural Network for Production Rescheduling Problem. In Proceedings of the 9th International Symposium on Integrated Uncertainty in Knowledge Modelling and Decision Making (IUKM), Ishikawa, Japan, 18–19 March 2022; pp. 279–290.
29. Kunapareddy, A.; Allaka, G. An Improved Genetic Algorithm for Production Planning and Scheduling Optimization Problem. In Proceedings of the International Conference on Intelligent Manufacturing and Energy Sustainability (ICIMES), Hyderabad, India, 21–22 June 2019; pp. 157–171.
30. Mao, C.L. PRODUCTION MANAGEMENT OF MULTI-OBJECTIVE FLEXIBLE JOB-SHOP BASED ON IMPROVED PSO. *Int. J. Simul. Model.* **2021**, *20*, 422–433. [CrossRef]
31. Shi, Y. Brain Storm Optimization Algorithm. In Proceedings of the 2nd International Conference on Swarm Intelligence (ICSI), Chongqing, China, 12–15 June 2011; pp. 303–309.
32. Cheng, S.; Qin, Q.; Chen, J.; Shi, Y. Brain storm optimization algorithm: A review. *Artif. Intell. Rev.* **2016**, *46*, 445–458. [CrossRef]
33. Zhao, F.; Hu, X.; Wang, L.; Xu, T.; Zhu, N.; Jonrinaldi. A brain storm optimization algorithm with feature information knowledge and learning mechanism. *Appl. Intell.* **2023**, *53*, 5199–5223. [CrossRef]
34. Agrawal, P.; Abutarboush, H.F.; Ganesh, T.; Mohamed, A.W. Metaheuristic Algorithms on Feature Selection: A Survey of One Decade of Research (2009–2019). *IEEE Access* **2021**, *9*, 26766–26791. [CrossRef]
35. Takahashi, K.; Fukuyama, Y.; Kawaguchi, S.; Sato, T. Optimal Production Scheduling using a Production Simulator and Multi-population Global-best Modified Brain Storm Optimization. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022.
36. Mafteiu-Scai, L. A New Approach for Solving Equations Systems Inspired from Brainstorming. *Int. J. New Comput. Archit. Their Appl.* **2015**, *5*, 10–18. [CrossRef]
37. Jordehi, A.R. Brainstorm optimisation algorithm (BSOA): An efficient algorithm for finding optimal location and setting of FACTS devices in electric power systems. *Int. J. Electr. Power Energy Syst.* **2015**, *69*, 48–57. [CrossRef]
38. Chen, J.; Cheng, S.; Chen, Y.; Xie, Y.; Shi, Y. Enhanced Brain Storm Optimization Algorithm for Wireless Sensor Networks Deployment. In *Proceedings, Part I, of the 6th International Conference on Advances in Swarm and Computational Intelligence*; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9140, pp. 373–381.
39. Hao, J.-H.; Li, J.-Q.; Du, Y.; Song, M.-X.; Duan, P.; Zhang, Y.-Y. Solving Distributed Hybrid Flowshop Scheduling Problems by a Hybrid Brain Storm Optimization Algorithm. *IEEE Access* **2019**, *7*, 66879–66894. [CrossRef]

40. Bhatt, A.; Dimri, P.; Aggarwal, A. Self-adaptive brainstorming for jobshop scheduling in multicloud environment. *Softw.-Pract. Exp.* **2020**, *50*, 1381–1398. [CrossRef]
41. Hou, Y.; Wang, H.; Fu, Y.; Gao, K.; Zhang, H. Multi-Objective brain storm optimization for integrated scheduling of distributed flow shop and distribution with maximal processing quality and minimal total weighted earliness and tardiness. *Comput. Ind. Eng.* **2023**, *179*, 109217. [CrossRef]
42. Qiu, H.; Duan, H. Receding horizon control for multiple UAV formation flight based on modified brain storm optimization. *Nonlinear Dyn.* **2014**, *78*, 1973–1988. [CrossRef]
43. e Oliviera, H.A., Jr. Homotopic quantum fuzzy adaptive simulated annealing [HQFASA]. *Inf. Sci.* **2024**, *669*, 120529. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.