**1. Introduction**

Generally, a classical resource-constrained project schedule only considers the precedence constraint between activities. While executing the schedule, some interference factors might influence the successors and require adjustments. It is complicated for a single constraint schedule to adjust when encountering resource contentions.

In recent years, the project scheduling problem with resource allocation has attracted the attention of many scholars. Kaiafa and Chassiakos (2015) suggested that existing resource allocation methods use strong resource constraints or duration limits. The authors divided the corresponding constraints into two kinds: “hard” and “soft.” In the optimised resource allocation model, the number of resources can fluctuate within a normal size.

Wiesemann, Kuhn, and Rustem (2012) divided resource allocation into two stages. A deterministic resource allocation model is established in the first stage, and uncertainty is added in the second stage to make the model more applicable to the stochastic scenario.

Aiming at the resource-constrained project scheduling problem with minimum and maximum time lags (RCPSP/max), Fu, Lau, and Varakantham (2015) designed the resource breakdown aware chaining procedure in the process of resource allocation based on three metrics: the mean, variance, and mean-variance heuristics. In addition, Almeida, Correia, and Saldanha-da-Gama (2016) studied one of the most recent extensions of RCPSP: the multiskill RCPSP. The authors applied the two new concepts of resource weight and activity grouping into the proposed heuristic algorithm to avoid random resource selection and allocation. They combined the eight activity priority rules to ensure that resources with higher weights can be allocated to activities with a shorter duration as much as possible.

Moutinho and Tereso (2014) proposed a mathematical model for the stochastic multimode RCPSP and used the discrete and finite partition of a timespan to evaluate the activity scheduling arrangement. According to the expected value of the required resources, the duration of each activity is as balanced as possible with that value to determine the activity start time and resource allocation mode to minimise the expected total project cost.

When Gülpınar, Çanakoğlu and Branke (2018) studied the stochastic multiperiod task-resource allocation problem, they assigned resources to various activities in the order of estimated marginal utility. They proposed forward (approximate) dynamic programming algorithms and an evolutionary algorithm.

Li et al. (2020) studied the decentralised multiproject time-cost trade-off problem, where project managers make resource allocation decisions and schedule projects in a distributed way. A multiagent-based cooperative approach with a negotiation protocol was proposed to mitigate resource contentions of autonomous local decisions and achieve a reasonable resource allocation for the overall decision-maker, with both exact and local-search-based heuristic algorithms developed.

Naegler and Schoenherr (1989) put forward the concept of the resource flow network in 1989. In addition, Leus and Herroelen (2001) used the so-called resource flow network to represent the transfer of resources across the activities of the project network schedule. After the resource allocation is concerned with project scheduling, the constraints in the schedule include precedence and resource constraints, and the resource flow network is established. The advantage of this network is that the new resource conflicts can be effectively prevented to make the schedule easier to adjust. The additional resource constraints are an effective measure to solve resource conflicts in the original network.

The research on project scheduling based on a resource flow network provides a new scheduling method. Solving resource conflicts is merely one of the functions of applying the resource flow network in project scheduling. As research goes further, the resource flow network could further apply to project scheduling, and building resource flow constraints could increasingly become a fundamental method in research. Some scholars have focused attention on research on the resource flow network and its application to project scheduling; however, the related literature is very limited so far.

The algorithms of the resource flow network have made some achievements. For example, Bowers (1995) defined the resource-constrained float as the total float for the CPM method based on the flow network. Artigues and Roubellat (2000) proposed a resource flow network, inserting new activities into the project using the network and constantly allocating resources. Aiming at the variation of activity duration, Leus (2004) and Leus and Herroelen (2004) put forward a dynamic resource allocation model for baseline schedule protection using the resource flow network and applied the branch and bound procedure to solve the resource allocation.

Artigues, Michelon and Reusser (2003) presented a resource flow network algorithm for allocating resources according to the activity, which is relatively simple. However, the algorithm adopts a direct allocation measure of randomly combined activity pairs rather than considering the precedence relation between activity pairs so that the consequent additional constraints are greater.

By generating a partial order schedule (POS) from the baseline schedule, Policella (2005) and Policella, Cesta, Oddi and Smith (2008) proposed chained resource flow network algorithms by allocating activities to the chains, including three algorithms (i.e., basic chaining, maximising common chains [MaxCC], and minimising interdependencies [MinID]). The difference lies in the different methods of selecting the initial chain, and the latter two algorithms are modified based on basic chaining.

In the study of project scheduling with discounted cash flow optimisation, some studies have introduced the resource flow network, and right-shift algorithms based on the resource flow network have been used to determine the time buffer to maximise the value of the target function. The importance of building a resource flow network is emphasised in analysing this cash flow optimisation problem. In these papers, the authors proposed two resource allocation algorithms, iterative sampling heuristic-unavoidable arcs (ISH-UA) and resource allocation with local search (RALS), which are compared with ISH, ISH2 and so on (Klimek & Łebkowski, 2013a, 2013b, 2014, 2015).

Deblaere, Demeulemeester, Herroelen and Van de Vonder (2007) proposed integer programming algorithms for the resource flow network based on three different objective functions, minimize the number of extra arcs (MinEA), maximize the sum of pairwise floats (MaxPF) and minimize the estimated disruption (MinED), and a heuristic algorithm for allocating resources called myopic activity-based optimisation (MABO). Among the three objective functions, MinEA refers to minimising the number of extra precedence relations (arcs) imposed by resource allocation, MaxPF refers to maximising the sum of pairwise floats, and MinED refers to minimising the estimated disruption between the realised and scheduled durations of the activities. While generating the resource flow network, MABO also performs in the order of the activities and minimises the stability cost (SC) of the schedule in resource allocation, building a project schedule with a certain robustness.

Based on the study on the RCPSP, considering the consumption of time and cost in the resource transfer process, Krüger and Scholl (2010) formulated a framework for resource transfer based on the following three points: the processing method of resource transfer, classification of resource transfer, and role of resources in the transfer process. They formulated a framework of resource transfer.

Klimek and Łebkowski (2011) defined a criterion for evaluating a resource flow network and proposed a robustness standard of resource allocation to measure the stability of the schedule. They deemed that the evaluation standard is more expedient than the existing robustness standards in the research.

Tritschler, Naber and Kolisch (2017) proposed a hybrid metaheuristic for the RCPSP with flexible resource profiles. This algorithm uses a flexible resource profile parallel schedule generation scheme and adopts the concept of delayed scheduling and nongreedy resource allocation to construct feasible schedules.

Nobuaki et al. (2017) developed a multistage dynamic scheduling model consisting of *N* classes of activities and a three-layer control structure. They proposed an activity selection and resource allocation method based on resource flows to maximise project profits.

Shi et al. (2020) studied the influence of the stability of the resource flow network on the performance of robust scheduling. They proposed the heuristic algorithm resource allocation based on forward activity priority and verified that it could reduce additional constraints and SCs through simulation experiments.

As the resource flow network becomes an important research method for project scheduling, its application has also made some gradual productive achievements. For example, Van de Vonder, Demeulemeester, Herroelen and Leus (2005) researched the ADFF method (scattered buffer methods of uncertain duration) and modified it by applying the resource flow network to solve the resource conflict issue. They implemented a comparison research with critical chain scheduling / buffer management in terms of the solution and quality robustness.

Based on the resource flow network, Van de Vonder, Demeulemeester and Herroelen (2008) analysed project scheduling with an uncertain duration and presented the three scattered buffer methods of resource flow dependent float factor, virtual activity duration extension and starting time criticality (STC) and compared the robustness of the three methods via experiments. The solution robustness of the STC method is better than that of the other two.

In their studies on the discrete-time and resource trade-off problem in railway scheduling, Tian and Demeulemeester (2010b, 2010a) introduced the concept of the resource flow network algorithm. They analysed the influence of various scheduling strategies on the average duration, standard deviation of the duration, timely project completion probability (TPCP), and SC of the schedule through simulation experiments.

Lambrechts, Demeulemeester and Herroelen (2011) studied the RCPSP. They proposed two surrogate robustness metrics based on the resource flow network, reflecting the delay of active duration caused by resource breakdowns under different scenarios. The authors used the Artigues algorithm to generate the resource flow network and considered that the Artigues algorithm takes less time than the MABO algorithm while guaranteeing the schedule stability.

Naber and Kolisch (2014) and Naber (2017) proposed four discrete-time models for the RCPSP with flexible resource profiles to determine the start time, resource profile and duration of each activity. The variable-intensity-based scheduling model has significant advantages in terms of solution quality and running time.

Cui and Liang (2018) presented a two-stage integrated optimisation algorithm to generate a robust project schedule. In the first stage, a myopic expected penalty cost algorithm builds a stable resource flow network by efficiently allocating resources to activities. In the second stage, an expected penalty cost algorithm that relies on the fixed resource flow network minimises the expected penalty cost by inserting time buffers before activities with higher delay risks.

Liang et al. (2020) studied robust project scheduling in the presence of activity duration variability and proposed a bi-objective optimisation model of the project due date and the total starting time criticality. Then, a two-stage heuristic algorithm deals with the resource allocation problem in the first stage and optimally determines the position and size of the time buffers with the simulated annealing algorithm in the second stage.

With the increasing research achievements concerning the resource flow network and its application, the generation algorithm of the resource flow network is proposed to improve its efficiency. This paper focuses on the research on the resource flow network algorithm and its implementation.

There are primarily two methods to allocate resources: taking advantage of the existing precedence relations between activity pairs in the original network and using the generated additional constraints between the activity pairs without precedence relations. Most resource flow network algorithms are sorted by activity order and adopt a sequential allocation method to deliver the resource capacity units to the successor activities, activity by activity or one to many.

However, some problems occur with these allocation algorithms. First, some algorithms adopt a mixed allocation method between activity pairs rather than differentiate between activity pairs with and without precedence relations while allocating resources. This method generates more constraints between activity pairs without precedence relations and cannot take advantage of precedence relations in delivering resources. Consequently, this method increases activity interdependencies and lowers schedule robustness.

Second, some algorithms consider precedence relations but have ineffective solutions to enhance the resource allocation capability. Most activity pairs with precedence relations have unsaturated resources allocated so that more resource capacity units are allocated to activity pairs without precedence relations, and more constraints are generated in the schedule.

In this paper, we propose a heuristic resource allocation algorithm to maximise the use of precedence relations (MaxPR). We design allocation processes for activity pairs with and without precedence relations. Moreover, we adopt saturated and time-lag span resource allocation strategies in the process for activity pairs with precedence relations to allocate more resource capacity units, reduce additional constraints, and lower the negative influence of additional constraints on schedule robustness.

Therefore, this paper contributes to the research in the following ways. First, this paper proposes a two-stage algorithm to allocate resources. In Stage 1, the activity pairs with precedence relations can be divided into zero-lag and time-lag relations to maximise the use of precedence relations in resource allocation. Based on the principle that the number of resources for distribution during project scheduling is equal to the resource availability, a reciprocating saturated resource allocation method is proposed for zero-lag activity pairs with precedence relations. In addition, a time-lag resource allocation discriminant and method are proposed for time-lag activity pairs with precedence relations.

By adopting these two allocation strategies, the resource units transferred to activity pairs with precedence relations can be maximised to reduce additional resource constraints. Second, the allocation strategy of unavoidable arcs is adopted to allocate the remaining resource units to the activity pairs without precedence relations in Stage 2, preventing additional constraints caused by random allocation resources to activity pairs without precedence relations and further reducing the number of additional constraints. Third, the simulation experiments compared with the other six algorithms and the case analysis proved that MaxPR and its multiple allocation strategies generate fewer constraints. It can also adapt to various network structures, demonstrating the concision and feasibility of this algorithm.

The structure and content of the paper are as follows. Section 2 introduces the basic definitions and notation of the resource flow network generated from the resource allocation of the baseline schedule and describes the optimised resource allocation model. Section 3 presents a literature review on the existing resource allocation algorithms. Section 4 elaborates on the proposed heuristic method of resource allocation, including resource allocation principles, allocation processes, allocation strategies, and algorithm pseudocode. Section 5 presents the computational analysis and comparison between the resource allocation algorithm proposed in this paper and the previously developed algorithms to verify the performance of the proposed algorithm and discuss the computational results. Section 6 analyses the case by applying the proposed algorithm. The final section presents the conclusions and limitations of the algorithm, along with a discussion on further research.

**2. Problem Description**

In a schedule, an activity allocates (or passes on) a used resource to another activity. The resource movement between this pair of activities can be regarded as a resource flow. The resources can be transferred between different pairs of activities; thus, the generated resource flow networks are also different, reducing the schedule robustness to varying degrees.

A resource flow network generated from resource allocation can be defined as a temporal graph  with resource flows between activities  and  in the baseline schedule. Moreover,  is the set of arcs  except for arcs  in the network. Supposing the number of activities in the baseline schedule is , the sequence number of activities is sorted from 0 to , and the duration of activity  is . The resource flow network has the same number of nodes as the baseline schedule network. In the network, the flow of the type  resources that transfer to arcs *A* and  is presented as, . Assuming the resource outflows of the dummy start activity 0 equal the resource inflows of the dummy finish activity , both flows are equal to the resource availability , (i.e., ). Besides the dummy start activity 0 and dummy finish activity *n*, the resource inflows of each activity must equal the resource outflows and resource requirements . The constraints of the resource flow network are , . Once the constraints are fulfilled, the network can be regarded as a feasible resource flow network.

In the project scheduling analysis, the resource allocation must be optimised between activity pairs to minimise the influence of additional resource constraints on the robustness. We can establish the following optimisation models:

  (1)

 s.t. ,  (2)

 ,  (3)

  ; (4)

In the above equations,  is the cost of deviating the time unit (also regarded as the robustness cost) when the realised start time of activity  deviates from the scheduled start time . In addition,  is the per-unit cost of deviating the time, including all kinds of management costs, coordination costs, and makespan costs when activity  deviates from the planned start time . Equation (1) represents the objective function of maximising the robustness, which means minimising the weighted sum of  and the deviation of  to minimise the influence of the resource allocation on project robustness. Equations (2) to (3) indicate the flow balance constraints for forming a feasible resource flow network. Equation (4) imposes integrity on the flow variables. We chose the number of additional resource constraints as the alternative objective function to perform optimisation in this paper. We also propose the heuristic algorithm MaxPR in resource allocation to minimise the number of generated additional constraints. The flow chart for this paper is illustrated in Figure 1.



Figure 1. Research flowchart

**3. Literature Review on Resource Allocation Algorithms**

***3.1. Artigues Algorithm***

Artigues et al. (2003) proposed a simple algorithm to generate a resource flow network by allocating resources to parallel activities at different times. The following code illustrates the algorithm steps.

Algorithm 1: Artigues algorithm

**for** increasing *t* in  **do**

 **for** to  **do**

 **if**  **then**

 **for** every resource type  **do**

 

 

 **while**  **do**

 **if**  **then**

 

 

 

 

 

 

This algorithm only generates a feasible resource flow network and does not consider the robustness of the optimising schedule or any other metrics.

***3.2. Chained Partial Order Schedule***

Policella et al. (2005) presented the algorithms of basic chaining, MaxCC and MinID, a set of chains representing the resource requirements of activities, generating the POS in a chained form and obtaining the resource flow network. The latter two algorithms are improved based on basic chaining. The only difference between the three algorithms is in the selection method of the activity chains. The specific steps of the basic chaining algorithm are presented in the following code.

Algorithm 2: Chaining procedure

Require: a problemand one of its fixed-time schedules 

Ensure: A partial order solution 

 

 Sort all activities according to the start times in 

 Initialise all chains as empty

 **for** all resources  **do**

 **for** all activity  **do**

 **for** 1 to  **do**

 Select chain 

 

 Add constraint 

 

**return** 

 

 (*a*) Project schedule (*b*) Basic chaining

Figure 2. Basic chaining and resource allocation.

According to basic chaining, for the project schedule in Figure 2(a), all activities in the baseline schedule are sorted by the start time, and the allocated order of the activities is . Taking the resource allocation of the sixth day as an example, when Activity 5 is allocated with the chain, suppose its initial chain is randomly chosen as Chain 14 so that the last activity in Chain 14 is Activity 1. Activity 5 requires two capacity units provided by Activity 1, and Activity 5 is allocated to two chains (i.e., Chains 13 and 14). Then, the next activity is Activity 6. Assuming Activity 6 is randomly allocated to Chain 12, which can be regarded as its initial chain, Activity 6 requires five capacity units; thus, Activity 6 is allocated to five chains, from Chain 8 to 12. However, the capacity units that Activity 6 requires are from Activities 1, 3 and 4. Similarly, resources can be allocated to Activities 8 and 7, and the chosen chains of Activity 8 are Chains 3 to 7. The capacity units for Activity 8 are provided by Activities 2 and 4. Activity 7 is allocated to Chain 2, and the resources are from the last activity (Activity 2) in Chain 2, as illustrated in Figure 2(*b*).

In basic chaining, Policella et al. (2005) defined an initial iterative procedure, , that allocates activities to available chains completely randomly. Hence, the chosen chains have great randomicity, increasing interdependencies between activities and lowering the schedule slackness. Moreover, MaxCC defines a set of common chains for two activities that require more resources based on basic chaining. While building the set of common chains, Policella et al. (2005) employed a heuristic chain selection procedure to arrange the two activities with the set of common chains to generate a schedule with slackness (i.e., maximising the number of common chains of the two activities to lower the additional constraints).

To this end, the following improvements were made to allocate an activity  in the MaxCC. An initial chain  was randomly selected from among those available for , and the constraint  is posted, where  is the last activity in chain . If  requires more than one resource capacity unit, then the remaining set of available chains can be divided into two subsets: the set of chains  with  as the last activity, and the set of chains  with . In addition,  is allocated first to chains of the first subset to satisfy all remaining resource requirements, . If the set is insufficient, the remaining units of  are allocated randomly to the first available chains of the second subset, .

 

(*a*) MaxCC (*b*) MinID

Figure 3. Maximising common chains and minimising interdependencies .

An improved algorithm of basic chaining, MaxCC, takes the measure of maximising the set of common chains for Activity 6 to reduce the additional constraints. We continue to take the project schedule in Figure 2(*a*) and resource allocation of the sixth day as examples. After completing the resource allocation of Activity 5, the next activity to be allocated is Activity 6. The initial chain of Activity 6 is randomly selected as Chain 5. Because Activity 6 requires five capacity units of resources, it is allocated to five chains, from Chains 5 to 9. Due to maximising the set of common chains to Activity 6, all required capacity units of Activity 6 are provided by Activity 4, which has precedence relations with it. Thus, no additional constraints are posted. Similarly, Activities 8 and 7 can be allocated to the remaining available chains, as illustrated in Figure 3(*a*).

While constructing the selection formulation of chains in MaxCC, if we select a chain for activity  randomly in the baseline schedule, the additional constraints between activities might be distributed to different chains (i.e., generating interdependencies of redundancy or synchronisation points), which enhances the interdependencies between activities and lowers the schedule slackness. In contrast, MinID improves the method of selecting the initial chain using a heuristic algorithm and considers the chains taken by activities with precedence relations.

Therefore, some improvements can be made to allocate an activity  in the MinID:

* 1. If the last activity  in chain  is the immediate predecessor activity of activity , chain  should be collected in set.
	2. If , chain is randomly chosen; if not, chain  is randomly chosen from the available chains.
	3. An additional constraint  is posted, where  is the last element of chain .

Figure 3(a) depicts the allocation results of MaxCC. We continue to take the project schedule in Figure 2(a) and resource allocation of the sixth day as examples. Activities 5, 8 and 7 do not consider whether a precedence relation exists between them and the last element of the chain when selecting the initial chain. The initial chain is selected randomly so that Activities 5, 8, and 7 have precedence relations with their immediate predecessors that distribute on different chains. Thus, additional constraints are posted between Activities 5, 8 and 7 and their last activity in the chain. Moreover, MinID improves the method of selecting the initial chain using the existing precedence relations to reduce the additional constraints. The initial chain of Activity 5 is Chain 3. Activity 5 requires two capacity units; thus, it is allocated to Chains 3 and 4. Similarly, the resource allocation results of other activities are presented in Figure 3(*b*).

***3.3. RALS and ISH-UA Algorithms***

Klimek and Łebkowski (2015) proposed the RALS algorithm. They created an  list in which the activities are sorted in increasing order by the respective start times in the baseline schedule , and the activities with the same start time are identified as a group. During the local search, an activity group is randomly selected at each iteration, and the order of activities within the group is rearranged. When allocating resources to activity  at time , priority is given to activities with arcs  connected to activity . These activities take the final position in the available resource chains at time , followed by activities that take the final position in the available resource chains and have no arc connected to activity . Due to rearranging the movement in the activity list  at each iteration, a resource allocation schedule with minimum additional arcs can be obtained by comparing it with the optimal allocation schedule. The pseudocode of the specific algorithm is presented below.

Algorithm 3: RALS algorithm

Identify the set of unavoidable arcs 

Create the list of activities sorted in increasing order by start times in ; if start times are the same, sort in decreasing order by aggregate resource demand;

Identify groups of activities with the same start time ;



**Repeat**



Clear the resource flow network  and set 

Add arcs from set  to set 

Randomly select an activity group from the list  for movement

For the selected group, rearrange the movement of this group in activity list ,

Create list 

**for**  **do**

**begin**

**for** ,  **do**

**begin**

Clear set 

take the final position in the available resource chain at time 

**while** **do**

take the final position in the available resource chain at time 

take the final position in the available resource chain at time 



**end**

**end**

**if**  **then**

**begin**

;

;

;

**end**

**until**

Return resource flow network 

In the RALS algorithm,  is the value of the assessment function for resource flow network,  is the largest current value of the assessment function,  denotes the best current resource flow network, and  represents the currently analysed resource flow network. In addition,  denotes the list of activities defining the order of allocation for which the resource flow network  is generated. Further,  is the set of activities used for construction resource flows for activities. Finally,  represents the number of iterations in which resource allocation is generated, and  is the number of current iterations.

Klimek and Łebkowski (2013b) proposed the ISH-UA algorithm, similar to the MinID algorithm illustrated in Section 3.2. The difference is that the algorithm considers the unavoidable arcs. When allocating resources to activity  at time , priority is given to the activities that take the final position in the available resource chains as the immediate predecessors of activity , followed by activities with unavoidable arcs connected to activity . The remaining steps of the ISH-UA algorithm are the same as those of the MinID algorithm.

**4. Resource Allocation Algorithms of MaxPR**

***4.1. Resource Allocation Principle***

In project scheduling, resources are allocated using two relations between the activity pairs: the existing precedence relations of the activity pairs and additional constraints generated from activities pairs. A greater number of additional constraints results in greater interdependencies between activities, which affects the schedule robustness.

In conventional resource allocation algorithms, the allocation order of activities usually depends on the time order of the random activity pairs rather than the precedence relations between pairs; thus, additional constraints are generated. For the MaxPR algorithm, to maximise the use of precedence relations, all activity pairs are separated into two categories: pairs with and without precedence relations. Resource allocation is also conducted in two stages: Stage 1 for pairs with relations and Stage 2 for pairs without relations, implemented right after Stage 1.

The precedence relations available for activity pairs in resource allocation can be divided into two categories: zero-lag relations and time-lag relations. In addition, MaxPR allocates resources as much as possible to a needed activity under a zero-lag relation. For time-lag relations, the total transferred resources should be limited to the resource availability. Thus, resource allocation is conducted considering the requirements of certain activities. The resource requirement balance of activity that has not been allocated yet must be considered. This conditional allocation method cannot guarantee resources for an activity. The discriminant of resource allocation is provided later in this paper.

However, not all activity pairs in the schedule have a precedence relation. Only partial activity pairs are allocated in Stage 1. Hence, the resource flow after this stage does not always meet the resource balance. Resources must be allocated to the remaining activity pairs with no precedence relations to achieve a resource flow balance in the schedule. In Stage 2, to reduce the number of additional constraints generated in this process, the resource allocation among unavoidable arcs without precedence relations must be given priority. Then, resources are allocated to other activity pairs without precedence relations.



Figure 4. Resource allocation principle.

***4.2. Resource Allocation Strategy***

*4.2.1. Saturated resource allocation*

The *stage* is the completion time (time point) for each activity sorted in ascending order in the baseline schedule. Several activities may have the same completion time in one stage. The *predecessor* and *successor* are the related activities for a certain finish and start time at a stage, and both may have one or more activities constituting the corresponding sets of activities. The *stage* *saturated resource number* refers to the maximum number of transferred resources  from all activity pairs with precedence relations at a certain allocation stage. The saturated number of resources in the stage can be calculated as follows:

 . （5）

In Equation (5), is the immediate predecessor set of the successor activity , representing the predecessor activity, is the number of immediate predecessor sets ,  is the number of successors, and  are the required capacity units (or remaining capacity units) of predecessor activity  and successor activity , respectively.

The saturated resource allocation means that the sum of transferred resources of all activity pairs with precedence relations at a certain stage equals the saturated resources in that stage. If the sum of transferred resources is lower than the saturated resources, it is called a nonsaturated resource allocation. Figures 5(*b*) and 5(*c*) display the allocation relations and the number of allocated capacity units by adopting different allocating methods for the activity pairs at Stage 3 (i.e., the nonsaturated resource allocation method and saturated resource allocation method with the allocated capacity units of six and eight, respectively).

Certain resource allocation strategies and algorithms are used in this paper to achieve saturated resource allocation, including an algorithm of searching single pairs of activities in an iterative cyclic manner. The specific procedures are illustrated as follows:

* Step 1: Search for a pair of activities with only one immediate predecessor starting from successor , taking the minimum capacity units of successor activity  and the immediate predecessor activity  (i.e., ) as the delivered allocated resources, deleting the allocated predecessor activities and successor activities with sufficient capacity units to ignore the resource allocation in these activities;
* Step 2: Search for a pair of activities with only one successor starting from predecessor activity , taking the minimum capacity units of predecessor activity  and the immediate successor activity (i.e., ) as the delivered allocated resources, deleting the allocated predecessors and successors with sufficient capacity units to ignore the resource allocation in these activities;
* Step 3: Repeat Steps 1 and 2 until no resources are available for allocation to the activity pairs so that the obtained saturated resources are  at this stage.

In the proposed algorithm, the allocation only applies to activity pairs with just one immediate predecessor or one immediate successor activity so that each activity pair has a unique allocation relationship. The allocated resources are minimum for that activity pair (i.e., ). No matter the predecessor or successor in the activity, pairs can use the precedence relations to maximise inflows or outflows according to the minimum resource capacity to achieve saturated resource allocation for each activity pair. Moreover, by adopting an iterative cyclic manner for searching for a single activity pair, the algorithm does not end until it finds all available activity pairs. Hence, the resource allocation in this stage is saturated.

   

(*c*) Saturated resource

allocation

(*a*) Baseline schedule

(*b*) Nonsaturated resource

allocation

Figure 5. Baseline schedule and resource allocation.

As illustrated in Figure 5(*c*), three successor activities are at Stage 3: Activities 5, 6 and 7. Starting from the successor, Activity 5, there is only one immediate predecessor, Activity 2. Activity 5 requires two capacity units, whereas Activity 2 has four capacity units. The minimum of two activities is two units, taking the minimum units as , which is the allocated number of resources for activity pair (2, 5). Because Activity 5 satisfies its resource requirements, we can no longer consider its resource allocation.

Next, we select the successor, Activity 7, with only one immediate predecessor, Activity 4. Activity 7 requires one capacity unit, whereas Activity 4 has six units. The minimum of two activities is one unit; therefore, the allocated number of resources for activity pair (4, 7) is . Activity 7 also satisfies its resource requirements; thus, we can put it aside. Then, starting from the predecessor, Activity 2, only one immediate successor exists, Activity 6. Activity 6 requires five capacity units, whereas Activity 2 merely has two remaining units. Thus, the number of resources allocated to activity pair (2, 6) is .

Similarly, the number of resources allocated to activity pair (4, 6) is . Hence, at Stage 3, the allocated resource between the predecessors and successors is units, and the number of saturated resources is  units, which means that Stage 3 achieves saturated resource allocation and reduces the additional schedule constraints posted between the predecessor and successor with precedence relations.

*4.2.2. Time-lag span resource allocation*

In delivering the resource process, resource allocation across a time lag is more intricate than that with zero-lag relations. Because the time-lag steps over at least one stage or even several stages, and the total number of delivered resources at each stage is less than the resource availability, we must determine whether activity pairs with time lag can deliver the resources directly and analyse the required capacity units of activities in the time lag. If the allocation is performed directly without analysis, it may imbalance the resource inflows and outflows of activities within the span of the time lag.

 

(*a*) Conventional resource allocation (*b*) Time-lag span resource allocation

Figure 6. Comparison of conventional and time-lag span resource allocations.

For time-lag span resource allocation, the aim is to estimate and determine the number of resources allocated to the time-lag span activity pairs. Three activity types affect the resource requirements in every stage of the time-lag span: activities with that stage as the completion time, activities that step over that stage, and time-lag span activity pairs with precedence relations. According to the principle that the number of resources in each stage of the plan is constant, the number of resources allocated to time-lag span activity pairs with precedence relations can be easily derived.

Assuming that activities  and  comprise a time-lag span activity pair with precedence relations, the completion time for  is stage , and the start time for  is stage ; thus, the number of resources allocated to the time-lag span activity pair  with precedence relations is :

 . (6)

In Equation (6),  is the number of activities completed at stage , denotes the resource capacity of activity  in the set of activities completed at stage . In addition,  is the number of activities whose start and completion times step over stage , and  is the resource capacity of activity  in the set of activities whose start and completion times step over stage . Further,  is the existing number of time-lag span activity pairs with precedence relations at stage , and  represents the number of resources allocated to time-lag span activity pairs with precedence relations for  time. Moreover,  and  are capacity units required by activities  and , respectively, and  denotes the resource availability of the schedule.

If the calculated resource allocated number , the time-lag span activity pair  with precedence relations can allocate capacity units to . Hence, the capacity units of predecessor activity  in activity pair  does not allocate resources to an activity with no precedence relations, avoiding generating additional constraints. When , the activity pair  cannot allocate capacity units due to resource availability.

Figure 6(*a*) illustrates that Activity 1 has precedence relations with Activities 2, 3 and 8. According to the conventional resource allocation strategy, due to the lack of analysis of time-lag span activity pairs with precedence relations, resource allocation has more to activity pairs without precedence relations, generating additional constraints.

As depicted in Figure 6(*b*), according to the time-lag span resource allocation strategy, while allocating the resources at Stage 2, Activities 3 and 9 are in Stages 2 and 4, respectively. The two activities combine to form an activity pair with precedence relations with a time-lag span, and the resource allocation of the activity pair (3,9) steps over Stages 3 and 4.

Two types of activities are allocated to resources at Stage 3. One type takes the stage as the completion time, such as Activities 2 and 4, and their required capacity units are  and . The other is a time-lag span activity pair (1,8) that requires the number of allocated resources .

In contrast, only one type of activity is allocated to resources at Stage 4, taking the stage as the completion time, including Activities 5, 6, 7 and 8. Their required capacity units are , , and , respectively. The remaining capacity provided by Activity 3 is , and the required capacity of Activity 9 is . According to Equation (6), to allocate resources to time-lag span activity pairs with precedence relations, we obtain ; thus, Activity 3 can allocate  capacity unit to Activity 9. Compared with Figure 6(*a*), we can determine from the allocation results in Figure 6(*b*) that far fewer additional constraints are posted after allocating time-lag span resources to activity pair (3, 9).

*4.2.3. Resource allocation with unavoidable arcs*

Deblaere, Demeulemeester, Herroelen and Van de Vonder (2006) proposed the concept of unavoidable resource arcs, referring to activity pairs that must generate a resource allocation relation caused by the balance of resource allocation in every stage for the scheduled activity pairs. The resource is no longer allocated to activities with precedence relations in Stage 2 of resource allocation. At this time, activity pairs in the set of unavoidable arcs in the whole project schedule generated in Stage 2 should all be activity pairs without precedence relations.

In Stage 2, determining unavoidable arcs in the schedule can be accomplished in steps. At each step, the following discriminant can be used to discriminate whether an unavoidable arc exists between activities  and.

 . (7)

When Equation (7) is established, the number of resources allocated to activities  and  is as follows:

 . (8)

In Equations (7) and (8),  refers to an activity set executed at time , and  denotes a set of activities in the baseline schedule whose starting time  meets the constraint . In addition, represents the remaining resource requirements of activity  in activity set ,  represents the resource requirements of activity  in activity set ,  represents the remaining resource supply of activity , and  represents the remaining resource requirements of activity. Additionally,  denotes the total sum of distributable resources at any time in the project schedule after Stage 1 of resource allocation. The limited number of resources within the project schedule in Stage 2 can be calculated and determined according to the remaining resource requirements of activities after Stage 1.

***4.3. Resource Allocation Procedures***

The proposed MaxPR algorithm can be divided into two stages. Stage 1 is the algorithm’s core, which is the key procedure to reduce additional constraints. By applying a sequential allocation method, stage by stage and activity by activity, the use of activity pairs with precedence relations is maximised to allocate more resources. However, Stage 1 is still an imbalanced allocation in terms of the overall resource flows in the schedule (i.e., the inflows of some activities are not equal to their outflows). Allocation only delivers to partial activities with precedence relations. As a result, on the allocation basis of Stage 1, it is necessary to conduct a complementary allocation to activities with imbalanced resource inflows and outflows in Stage 2. The balanced allocation in Stage 2 primarily applies to the activity pairs without precedence relations; thus, additional constraints may be generated in this process. The pseudocode of this algorithm is illustrated as follows.

Algorithm 4: Maximising precedence relations for resource allocation

**begin**

 /\* stage 1 \*/

 Determine time instants by activity completion time

 **for**  to  **do**

 

 

 

 **for**  **do**

 **while** =1 **do**

 

 

 

 **for**  **do**

 **while** =1 **do**

 

 

 

 **if**  **then**

 Calculate the number of resources allocated to a time-lag span

 /\* stage 2 \*/

 **for**  to  **do**

 **for**  **do**

 **for**  **do**

 **if**  **then**

 

 **for**  **do**

 

 **for**  **do**

 **while**  **do**

 

**end**

In Algorithm 4, denotes the start time of stage , is the activity set of the immediate predecessor, represents the activity set of the immediate successor, and is the predecessor activity set with a precedence relation with the successor in stage . In addition, denotes the successor activity set with a zero-lag precedence relation with the predecessor in stage , and is the successor activity set with a time-lag span precedence relation with the predecessor in stage . Further,  represents the capacity unit to which predecessor activity  can allocate, and  denotes the needed capacity units for successor activity . Moreover,  represents the allocated number for activity pair ,  is a set of activities whose starting time is  in stage , and is a set of predecessors of the activitywhose starting time is . Furthermore,  denotes the remaining resource demand of activity  in activity set,  is the remaining resource demand of activity , and  is the remaining resource demand of activity. Additionally,  is the limit of the resources within the project duration in the second process. Next,  denotes the activity set sorted according to the completion time for all activities in ascending order, and  is the activity set of successor, and the start time of every activity in  is greater than the finish time for activity .

**5. Experimental Evaluation**

To elaborate on the feasibility and rationality of the algorithm proposed in this paper, we use a comparative analysis approach to evaluate MaxPR and the other six algorithms (i.e., the Artigues algorithm, basic chaining, MaxCC, MinID, RALS and ISH-UA).

***5.1. Computational Data and Evaluation Indicators***

The computational procedures were implemented in MATLAB (R2014a). They were run on a CORE i7 2.40 GHz computer with 16 GB of RAM using the Window 8.1 operating system.

To analyse the performance of each algorithm under the conditions of different project scales and uncertainty levels, we selected four data sets, J10, J30, J60 and J90, each consisting of 200 project cases. All computational data were derived from ProGen (Kolisch, Sprecher, & Drexl, 1995), an instance generator for solving project scheduling problems. In the computational experiment, the random variable of uncertain duration for each set of activities was assumed to follow the lognormal probability distribution. Considering the four levels of uncertainty, we can set the corresponding standard deviation of duration as  =  0.1，0.3，0.6 and 0.9 and use the given duration of the project as the expected duration . We applied the function in MATLAB to calculate the random duration  that follows the lognormal probability distribution in each activity set. The activity weight  indicates the importance of the activity. Assuming  follows a discrete triangular distribution, it is calculated using the equation where . The number of simulation experiments is 1000.

This paper employs evaluation metrics reflecting the solution and quality robustness and the comprehensive metrics of robustness, including the SC, TPCP, average project length (APL), and the number of precedence constraints posted (NPCs) to evaluate the robustness of resource allocation schemes generated by different algorithms.

***5.2. Analysis of Computational Results***

This paper analyses the influence of different algorithms on robustness from the aspects of the project scales, uncertainty levels, and adaptability of the algorithms.

*5.2.1. Influence of project scale on robustness*

Table 1 lists the computational results of four data sets: J10, J30, J60 and J90, using seven algorithms, with the standard deviation of the duration .

Table 1. Influence of project scale on robustness

|  |  |  |
| --- | --- | --- |
|  | J10 | J30 |
| SC | TPCP | APL | NPC | SC | TPCP | APL | NPC |
| Artigues | 69.94 | 0.46926 | 34.95 | 6.92 | 624.72 | 0.11987 | 80.63 | 31.64 |
| Basic chaining | 69.68 | 0.47017 | 34.92 | 7.57 | 632.15 | 0.11631 | 80.83 | 33.11 |
| MaxCC | 69.55 | 0.46994 | 34.92 | 7.04 | 616.07 | 0.12131 | 80.50 | 30.42 |
| MinID | 68.16 | 0.47757 | 34.77 | 7.39 | 584.67 | 0.13458 | 79.72 | 29.51 |
| RALS | 68.71 | 0.47352 | 34.79 | 7.43 | 585.23 | 0.13087 | 80.14 | 29.83 |
| ISH-UA | 69.35 | 0.47018 | 34.91 | 7.28 | 618.78 | 0.12496 | 80.97 | 30.79 |
| MaxPR | 68.10 | 0.47826 | 34.74 | 5.26 | 581.10 | 0.13661 | 79.61 | 25.78 |
|  | J60 | J90 |
| SC | TPCP | APL | NPC | SC | TPCP | APL | NPC |
| Artigues | 2857.64 | 0.01398 | 130.75 | 71.38 | 7576.35 | 0.00148 | 171.01 | 113.95 |
| Basic chaining | 2987.86 | 0.01153 | 131.97 | 77.18 | 8015.62 | 0.00107 | 173.41 | 126.45 |
| MaxCC | 2856.08 | 0.01403 | 130.82 | 69.82 | 7647.58 | 0.00133 | 171.57 | 113.15 |
| MinID | 2677.71 | 0.01871 | 129.05 | 65.98 | 7201.27 | 0.00254 | 168.78 | 106.17 |
| RALS | 2695.13 | 0.01723 | 129.87 | 67.24 | 7356.39 | 0.00247 | 169.56 | 108.32 |
| ISH-UA | 2879.24 | 0.01637 | 130.95 | 71.82 | 7743.76 | 0.00189 | 172.87 | 114.98 |
| MaxPR | 2599.17 | 0.02080 | 128.24 | 59.41 | 6844.39 | 0.00335 | 167.02 | 94.43 |

As the results in Table 1 reveal, for the larger scale of benchmarks J30, J60 and J90, MaxPR is still superior to the other algorithms with the expansion of the project scale. To illustrate this advantage, we compared MaxPR and basic chaining and analysed the computational results in J30, J60 and J90.

With the expansion of the project scale, by applying MaxPR, the SC reduces by 8.08%, 13.00% and 14.61% in J30, J60 and J90, respectively, compared with basic chaining. The reduction of the SC demonstrates that applying MaxPR to resource allocation improves the solution robustness and stability of the schedule compared with the other algorithms. In addition, MaxPR is more suitable for large-scale project resource allocation than the others.

For the TPCP, the completion probability of MaxPR increases by 17.45%, 80.40% and 213.08% in J30, J60 and J90, respectively, compared with basic chaining. A larger project scale results in an increased completion probability. Therefore, MaxPR can improve the quality robustness of the project to some extent.

The APL also reflects the quality robustness of the schedule, like the TPCP. The computational results in Table 1 indicate that the makespan of MaxPR shortens by 1.51%, 2.83% and 3.69% in J30, J60 and J90, respectively, compared with basic chaining. Thus, APL is more appropriate for improving the quality robustness of large-scale projects than the others.

For NPC, with the expansion of the project scale, MaxPR generates a lower absolute number of additional constraints, the NPC decreases by 22.14%, 23.02% and 25.32%, respectively, compared with basic chaining. The decreasing tendency indicates that a larger project scale results in more advantages for MaxPR. In other words, a smaller NPC with more precedence relations can be taken advantage of in resource allocation by applying this algorithm, reducing the additional constraints.

*5.2.2. Influence of different uncertainty levels of activities on robustness*

We selected results from J60 to analyse the practicability of the algorithms in a large-scale project. Table 2 lists the computational results of applying the seven algorithms in J60 when the standard deviation of the duration is 0.1, 0.3, 0.6 and 0.9.

Table 2. Influence of different uncertainty levels of activities on robustness

|  |  |  |
| --- | --- | --- |
| J60 |  |  |
| SC | TPCP | APL | NPC | SC | TPCP | APL | NPC |
| Artigues | 338.51 | 0.05926 | 101.20 | 71.48 | 1235.58 | 0.01781 | 112.06 | 71.94 |
| Basic chaining | 377.64 | 0.04052 | 101.62 | 77.15 | 1320.19 | 0.01319 | 112.87 | 77.84 |
| MaxCC | 365.98 | 0.04518 | 101.51 | 70.32 | 1260.81 | 0.01582 | 112.36 | 70.27 |
| MinID | 331.57 | 0.06323 | 101.14 | 66.12 | 1172.06 | 0.02313 | 111.37 | 66. 50 |
| RALS | 334.65 | 0.06475 | 101.25 | 67.75 | 1198.23 | 0.02387 | 111.74 | 67.81 |
| ISH-UA | 369.87 | 0.05234 | 101.87 | 71.13 | 1287.45 | 0.01921 | 112.56 | 71.93 |
| MaxPR | 316.58 | 0.07122 | 100.99 | 59.41 | 1123.35 | 0.02743 | 110.88 | 59.65 |
| J60 |  |  |
| SC | TPCP | APL | NPC | SC | TPCP | APL | NPC |
| Artigues | 2857.64 | 0.01398 | 130.75 | 71.38 | 4730.08 | 0.02014 | 152.82 | 71.38 |
| Basic chaining | 2987.86 | 0.01153 | 131.97 | 77.18 | 4891.25 | 0.01738 | 154.24 | 77.18 |
| MaxCC | 2856.08 | 0.01403 | 130.82 | 69.82 | 4690.59 | 0.02031 | 152.56 | 69.82 |
| MinID | 2677.71 | 0.01871 | 129.05 | 65.98 | 4432.18 | 0.02561 | 150.13 | 65.98 |
| RALS | 2695.13 | 0.01723 | 129.87 | 67.24 | 4587.25 | 0.02687 | 151.56 | 67.25 |
| ISH-UA | 2879.24 | 0.01637 | 130.95 | 71.82 | 4716.43 | 0.02243 | 153.08 | 71.83 |
| MaxPR | 2599.17 | 0.02080 | 128.24 | 59.41 | 4324.76 | 0.02808 | 149.04 | 59.41 |

With the increasing uncertainty of the duration, its standard deviation increases from to , and the SC of the simulation experiment for each algorithm also increases gradually. However, MaxPR continues to have a minimal SC, which demonstrates that MaxPR has certain advantages in improving the solution robustness of the project. This algorithm is also suitable for a resource allocation scheme with certain requirements for the solution robustness in the schedule.

For the TPCP, with the increasing uncertainty of the duration, the TPCP of MaxPR is 75.77%, 51.91%, 80.40% and 61.57%, revealing a fluctuating result without an increasing or decreasing tendency. The simulated TPCP results of the algorithms are not high, regardless of whether MaxPR or other algorithms are used; however, the TPCP of MaxPR is higher than that of the others. Thus, MaxPR still enjoys certain advantages in the quality robustness of the project.

For the APL, with the increasing uncertainty of the duration, the simulated makespan of MaxPR is less than that of basic chaining, although the absolute value of each makespan is prolonged. In contrast, MaxPR has an increasing tendency of a reduced makespan, the APL decreased by 0.62%, 1.79%, 2.83% and 3.37%, respectively, compared with basic chaining, indicating that MaxPR can take full advantage of precedence relations in resource allocation. When the uncertainty of the duration increases, the critical path and makespan of the project is not prolonged too much, which also accounts for some advantages in improving the schedule robustness compared with other algorithms.

For the NPCs, MaxPR has fewer additional constraints than basic chaining at different standard deviations of the duration , revealing that MaxPR for resource allocation can effectively reduce the number of additional constraints. In contrast, the other algorithms have additional constraints due to the insufficient use of precedence relations.

**6. Case Analysis**

A 110 kV substation construction project is used as an example of the resource allocation analysis to verify the practicality of the proposed MaxPR algorithm. The substation is in the economic and technological development zone of Yuncheng City, Shanxi Province, China. It is an unattended substation relying on intelligent substation technology and integrated automation technology.

According to the construction organisation and substation requirements, the project should first be divided into several activities, determining the logical relations and workflow between activities and calculating the completion time for each activity on the grounds of the quantities and resources invested in each activity. The basic data for substation project scheduling are listed in Table 3.

Table 3. Basic data for project scheduling

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Activity | Predecessor activity | Successor activity | Time(day) | Resource(labour) |
| 1 | Project start | ─ | 2 | 0 | 0 |
| 2 | Construction of temporary building | 1 | 3,4,5,6,7 | 14 | 12 |
| 3 | Installation of main transformer structure | 2 | 8 | 12 | 10 |
| 4 | Civil construction of 110 kV main building | 2 | 9 | 10 | 6 |
| 5 | Civil construction of other buildings | 2 | 9 | 14 | 8 |
| 6 | Civil construction of control room | 2 | 10,11 | 17 | 10 |
| 7 | Road hardening | 2 | 16 | 8 | 5 |
| 8 | Installation of main transformer | 3 | 12 | 12 | 12 |
| 9 | Installation of cables and capacitors | 4,5 | 12 | 10 | 10 |
| 10 | Installation of protection screen | 6 | 13 | 15 | 6 |
| 11 | Installation of AC and DC systems | 6 | 14 | 3 | 5 |
| 12 | Laying out secondary cables and grounding devices | 8,9 | 15 | 10 | 9 |
| 13 | Installation of communication and automation system | 10 | 15 | 12 | 10 |
| 14 | Installation of lighting and fireproof system | 11 | 16 | 7 | 8 |
| 15 | Connection of secondary wiring | 12,13 | 16 | 9 | 8 |
| 16 | Commissioning | 7,14,15 | 17 | 12 | 12 |
| 17 | Final acceptance | 16 | 18 | 6 | 8 |
| 18 | Commercial operation | 17 | 19 | 4 | 6 |
| 19 | Project completion | 18 | ─ | 0 | 0 |

The project was scheduled to officially start on July 1, 2019. The resource flow network of the project is illustrated in Figure 7, and the baseline schedule of the project calculated using the tabu search algorithm is presented in Figure 8, with a resource availability of 20 labourers and a duration of 102 days.



Figure 7. Resource flow network of the project.



Figure 8. Baseline project schedule.

In project scheduling, Activities 1 and 19 are dummy activities. We set the resource requirements to 20, which equals the resource availability while allocating resources. Based on the baseline project schedule, the MaxPR resource allocation algorithm is adopted to build the resource flow network (Figs. 9 and 10). In Stage 1, the zero-lag and time-lag span allocation strategies are used to allocate resources to activity pairs with precedence relations, and the total number of such activity pairs is 16. In Stage 2, resource allocation with unavoidable arcs is performed, and resources are randomly allocated to subsequent activities one by one according to the start time, resulting in 16 additional constraints (represented by dashed lines with arrows). Therefore, this algorithm maximises precedence relations in generating the resource flow network to reduce the number of additional constraints and lower the influence of resource constraints on schedule robustness.



Figure 9. Resource allocation using the MaxPR algorithm.



Figure 10. Resource flow network using the MaxPR algorithm.

**7. Conclusions and Further Discussion**

Different resource allocation methods generate different resource flow networks and have different effects on schedule robustness. This paper proposes a resource allocation algorithm of MaxPR based on the latest research results of resource allocation. The proposed algorithm has clear principles, easy-to-operate procedures, and simple calculations that can be performed in a brief time, generating a concise resource flow network with fewer additional resource constraints. The simulation experiments prove that the algorithm is feasible for different resource allocation schedules with diverse scales and uncertainties.

The proposed MaxPR algorithm optimises resource allocation in two stages. However, it still adopts a single allocation method, simply overlaying the allocation schemes from two stages to generate the final resource allocation schedule. There might be a variety of resource allocation schemes in Stage 1. Although the resource allocation schedule has approaching minimum additional constraints generated from Stage 2 (based on the single allocation scheme in Stage 1), the final resource allocation schedule cannot be guaranteed to have the least additional constraints. Therefore, the algorithm still has shortcomings that require further discussion.

Generally, constructing a resource flow network in a baseline schedule reduces schedule robustness and has a complicated effect on the robustness. This paper presents the study of the approach that influences the robustness of the resource flow network by establishing the resource flow network first and then generating a robust resource allocation schedule. This method is reasonable, taking the establishment of the resource flow network and the generation of robust scheduling as the overall research object to build a resource flow network with minimum influence on schedule robustness, which is also an interesting topic worthy of further study.