





Article

Complicated Time-Constrained Project Scheduling Problems in Water Conservancy Construction

Song Zhang ^{1,*}, Xiaokang Song ¹, Liang Shen ¹ and Lichun Xu ²¹ School of Management, Xuzhou Medical University, Xuzhou 221004, China² School of Public Health, Xuzhou Medical University, Xuzhou 221004, China

* Correspondence: szhang@xzhmu.edu.cn

Abstract: Water conservancy project scheduling is an extension to the classic resource-constrained project scheduling problem (RCPSP). It is limited by special time constraints called “forbidden time windows” during which certain activities cannot be executed. To address this issue, a specific RCPSP model is proposed, and an approach is designated for it which incorporates both a priority rule-based heuristic algorithm to obtain an acceptable solution, and a hybrid genetic algorithm to further improve the quality of the solution. In the genetic algorithm, we introduce a new crossover operator for the forbidden time window and adopt double justification and elitism strategies. Finally, we conduct simulated experiments on a project scheduling problem library to compare the proposed algorithm with other priority-rule based heuristics, and the results demonstrate the superiority of our algorithm.

Keywords: resource-constrained; project scheduling; genetic algorithm; water conservancy



Citation: Zhang, S.; Song, X.; Shen, L.; Xu, L. Complicated Time-Constrained Project Scheduling Problems in Water Conservancy Construction. *Processes* **2023**, *11*, 1110. <https://doi.org/10.3390/pr11041110>

Academic Editors: Luis Puigjaner and Vladimir Mahalec

Received: 6 March 2023

Revised: 26 March 2023

Accepted: 30 March 2023

Published: 5 April 2023



Copyright: © 2023 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 (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Water conservancy construction usually endures a long cycle of multi-tasks, absorbing substantial investments of money and recourses; it is therefore difficult to rationally plan and scientifically schedule. Traditional scheduling approaches which have widely served water conservancy projects since the 1950's have been phased out, such as Gantt chart, critical path method (CPM), and program evaluation and review technique (PERT). An effective alternative, independent from human experiences, is much needed.

Water conservancy project scheduling is a resource-constrained project scheduling problem (RCPSP); an optimization task with time and resource limitations between each activity. RCPSPs have been extensively studied for decades and many scholars have presented insightful reviews on them from different perspectives [1–7]. Water conservancy projects differ from classical RCPSPs in that they are subject to special time constraints.

There are approximately three categories of RCPSP research related to time constraints. In the early days, a typical time-constrained extension to RCPSP was RCPSP/Max, referring to an RCPSP with minimum/maximum time lags; this was elaborated in Neumann et al.'s [8] review. Ismael de Azevedo et al. [9] proposed a satisfiability and workload-based exact method to deal with the resource-constrained project scheduling problem with generalized precedence constraints (RCPSP/Max). Morin et al. [10] studied an original variant of the RCPSP, the PARCPSP. While the start and completion times of the activities can be arbitrary moments in time the limitations on resource usage are considered on average over aggregated periods of parameterized length. This kind of problem only considers the relative time interval between activities, which is general. The second type is the periodic time window problem. Chen et al. [11] introduced two time-related constraints: the time window constraint, outlining each activity's execution time interval; and the time-schedule constraint, defining the pre-specified beginning time for certain activities, such as train schedules. Zhan et al. [12] and Franck et al. [13] adopted calendars to differentiate time

intervals, such as workdays and non-workdays. Then they sorted activities into interruptible and uninterruptible categories and arranged them according to calendar limits. Similarly, Yang et al. [14] proposed another constraint, called a time-switch constraint, which designates a certain time interval within a cycle specifically for starting an activity. The third category considers the special time constraints of the activities in the scheduling problem, making the problem more practically significant and gradually becoming a research hotspot. In recent years, more and more work has been focused on the time window problem of the RCPSP, which exists in many fields in real life, such as course scheduling, vehicle routing planning, and human resource scheduling. Bomsdorf et al. [15] proposed a model to solve a movie shooting scheduling problem. Lorenzoni et al. [16] found that ships attending a port within a limited time constituted a multi-mode resource-constrained scheduling problem; they solved this using mathematical algorithms. Vanhoucke et al. [17] proposed the concept of “quality-dependent time slots” and assigned them to activities to achieve a minimal loss of quality. They also successfully scheduled a biological R&D project using these time slots. There are two kinds of time window constraints in railway transportation, one is soft time window and the other is hard time window. The soft time window requires goods to arrive within the time window to the greatest possible extent, with penalties if they are violated. The hard time window requires that the goods must arrive within the time window, otherwise the goods will be rejected. Mi et al. [18] considered the mixed time windows of the multimodal transportation optimal routing model and used the CSO algorithm to solve the problem. Drexler et al. [19] designed a problem instance generator that integrates novel constraints named “forbidden periods” to allocate an earliest/latest pair of ending times to each activity. It can be used for many labor-related scheduling problems, such as course scheduling where two lessons of the same course cannot be assigned in two periods at the end of one day and at the beginning of the next.

In summary, scholars have already done a lot of research on the time constraints of RCPSPs. However, most of the previous time-related studies have focused on the time constraints between individual activities within projects and have seldom noticed that real-life projects are vulnerable to outside time limitations. While the start time of the project is random, and some activities in the project can't be carried out in certain time periods, which may affect the overall progress of the project. For example, there are environmental work windows in dredging fleet scheduling [20], weather windows in offshore operation [21], and special time windows in water conservancy construction, additionally, the project is affected by seasonal conditions and environmental regulations [22]. For example, the construction of river dikes and channels, which usually lasts for more than one year, is subject to seasonal flooding. Some activities can be carried out during floods while others cannot; this makes scheduling these activities even more complicated. Scholars refer to these special time intervals as “forbidden time windows” and have applied them to many construction project scheduling problems. Blazewicz et al. [23] have shown that the RCPSP is an NP-hard problem and that forbidden time windows can impose extra difficulties and complications in finding its solution.

To address the challenges of forbidden time window problems, we can convert them into a type of RCPSP, taking the shortest project duration as the goal and combining heuristic algorithms and meta-heuristic algorithms to solve large-scale project scheduling problems. Our major contributions are as follows:

- (1) A mathematical model of water conservancy construction scheduling aiming at the constraint of forbidden time windows was established, and a simulation project containing 30, 60, 90, and 120 tasks was generated. We separately combined the parallel scheduling scheme and the serial scheduling scheme with the seven different priority rules and applied them to all the instances of the four types; we discussed which kind of rules perform better.
- (2) A hybrid genetic algorithm was designed, which took the resource utilization rate in the forbidden time windows as the basis for cross operation. Finally, the rule-based heuristic algorithm was compared with a genetic algorithm.

The remaining structure of this paper is as follows: in Section 2, we introduce the related problems; in Section 3, we propose the solution; in Section 4, the simulation experiment is presented; and finally, in Section 5, we present the conclusions and our planned future work.

2. Problem Description

A typical RCPSP can be described as follows. A project is composed of a set J^+ that contains J activities, where the two dummy activities, $j = 1$ and $j = J$, represent the project beginning and the project termination, respectively. The dummy activities take zero time and recourse. Activity j may last d_j time duration: starting from s_j and ending at c_j . Obviously, $s_j + d_j \leq c_j$. Each activity cannot be interrupted during its exertion and all activities are ordered by precedence constraints: $Pred(j)$ denotes the set of the activities that immediately precede j , which means activity j cannot start until all the activities in $Pred(j)$ end; similarly, $Succ(j)$ is the set of the activities that immediately succeed activity j , which means only at the time of or after activity j end, can activities in $Succ(j)$ begin. The project can allocate K recourses. R_k represent the total amount of recourse k in a cycle and r_k the amount of recourse k that activity j requires throughout its execution. At any time, r_k must be lesser than R_k . All of the parameters listed above are non-negative integrals. Thus, a RCPSP is a combinatorial optimization problem that aims to find the minimum makespan with constraints of precedence and resources.

Water conservancy projects are constrained by “forbidden time windows”. During a forbidden time window, certain special activities cannot be executed. They must be either postponed or conducted in advance. Figure 1 depicts an example: a project consists of 21 activities, including two dummy ones. It only needs one recourse R , and its total units is four. Special activities, denoted by boxes in Figure 1, are $\{2, 7, 11, 15, 18, 19\}$ and a forbidden time window stretches between $[5, 16]$. Figure 2 illustrates one schedule to this project. Activity 2 cannot begin at time point 15, because it is “forbidden” during the time interval $[5, 16]$, which reduces the efficiency of the whole project. Figure 3 provides a better solution; activity 2 is brought forward, avoiding the “forbidden time window”, which reduces the working period of the project.

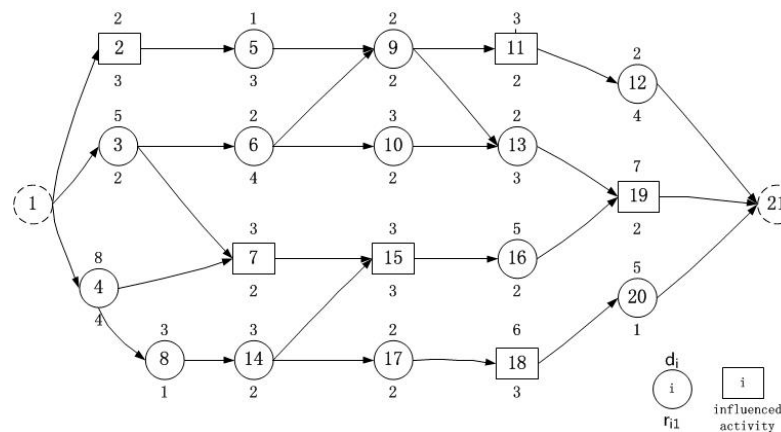


Figure 1. Sample project.

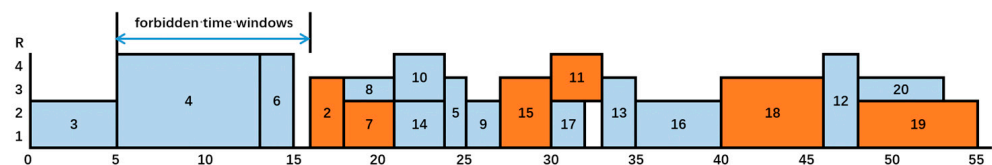


Figure 2. Schedule 1.

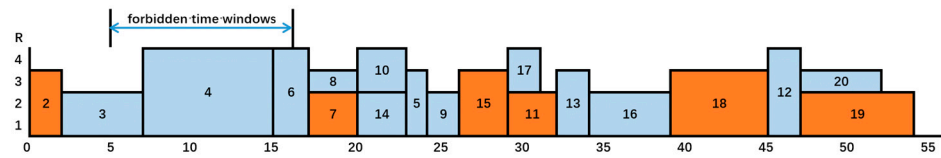


Figure 3. Schedule 2.

In practice, a project may involve multiple forbidden time windows, due to the weather or to technical problems. For better generalization, we regard $[ST_{i1}, ST_{i2}]$ as a forbidden time window, and U^+ as the set of special activities. Thus, a RCPSP model with forbidden time windows can be described as follows.

$$\text{Min } s_J \tag{1}$$

s.t.

$$s_j - s_i \geq d_i, j \in J^+, i \in \text{Pred}(j); \tag{2}$$

$$\sum_{j \in A(t)} r_{jk} \leq R_k, k \in K, t \geq 0; \tag{3}$$

$$s_j > ST_{j2} \text{ or } s_j + d_j < ST_{j1}, j \in U^+, U^+ \in J^+; \tag{4}$$

where Equation (1) signifies that the optimization goal of the model is the minimum makespan; Equation (2) gives the precedence constraints between activities; Equation (3) gives the recourse constraints; while Equation (4) indicates that special activities must be either completed before the special time intervals or started after them. When $U^+ = \emptyset$ this model changes back to a classic RCPSP. If all the activities correspond to the same forbidden time window, this model transforms into a calendar-constrained problem. Table 1 lists all the parameters used in the model.

Table 1. Parameters used in the proposed model.

| Parameter | Meaning |
|----------------------|--|
| j | activity number, $j = 1, 2, \dots, J$ where J is the total number of activities |
| d_j | duration of activity j |
| s_j | starting time of activity j |
| c_j | ending time of activity j |
| $\text{Pred}(j)$ | the set of activities immediately preceding activity j |
| $\text{Succ}(j)$ | the set of activities immediately succeeding activity j |
| U^+ | the set of special activities |
| J^+ | the set of all the activities |
| t | time number, $t = 1, 2, \dots, T$; where T is the deadline of the whole project. |
| A_t | the set of activities that are being executed at time t , $A_t = \{j \mid j \in J^+ \cap s_j \leq t \leq s_j + d_j\}$ |
| $[ST_{i1}, ST_{i2}]$ | the forbidden window that constrains activity i |
| k | recourse number, $k = 1, 2, \dots, K$; where K is the total number of types of recourses. |
| R_k | the amount of recourse k |
| r_k | the amount of recourse k that activity j requires |

3. Solution

The special-time-constrained RCPSP enforces critical barriers when scheduling special activities. We propose a tailored priority-based heuristic algorithm and a hybrid genetic algorithm to tackle it.

3.1. Priority Rule-Based Heuristic Algorithm

Heuristics have been widely accepted in RCPSPs, especially in those of large scales, thanks to their fast speed. A higher-level of heuristics is the metaheuristic algorithm. Its high-efficiency and simple logic has boosted its applications around business plans and software pipelining. A typical metaheuristic algorithm consists of two critical procedures: schedule generation scheme (SGS), and priority rule [24]. Two types of SGS are feasible in practice: the serial SGS based on activity-incrimination, and the parallel SGS based on time-incrimination. Priority rules are used to deploy one or both SGS(s) and to finally find the best solution to the scheduling problem.

Priority-based heuristics are satisfactory at tackling classic RCPSPs but would be inept at the special one discussed in this paper. Therefore, we can tailor the priority rules to forbidden time windows using the following procedure. If a special activity is one of the options and can be finished before its forbidden time begins, it will be selected. If two or more special activities are among the options and they can all be completed before the forbidden time windows, which means that they are all tied, the activity with the smallest label will be chosen. This is the tie-breaking rule.

Table 2 lists both schedule generation schemes and their corresponding priority rules adopted in this paper.

Table 2. Schedule generation schemes and corresponding priority rules.

| Serial Scheduling Generation Scheme | Parallel Scheduling Generation Scheme |
|-------------------------------------|---------------------------------------|
| most immediate successors, MTS | most immediate successors, MTS |
| total resource demand, TRD | total resource demand, TRD |
| shortest processing time, SPT | shortest processing time, SPT |
| latest starting time, LST | latest starting time, LST |
| minimum slack time, MST | minimum slack time, MST |
| critical activity, CA | critical activity, CA |
| latest finishing time, LFT | worst case slack, WCS |

3.2. Hybrid Genetic Algorithm

Hartmann [25] proposed an effective genetic algorithm for RCPSP. We have borrowed the principles of this algorithm, and further designed specific crossover operators for forbidden time windows, as well as adopting double justification and elite selection strategy. Our hybrid genetic algorithm could provide final solutions of an improved quality.

The Algorithm 1 proceeds as follows.

Algorithm 1: Hybrid genetic algorithm

Step 1; initialization: define popsize, Gen, and Pm
 Step 2; initiate population pop
 Step 3; double-justify chromosomes in pop
 Step 4; calculate fitness of each individual
 Step 5; select parent
 Step 6; use the crossover operator for forbidden time windows
 Step 7; exert exchange mutilation
 Step 8; generate new population
 If the maximum number of iterations are reached end the algorithm
 otherwise
 go to Step 3.

First, the algorithm sets that the initial population contains popsize individuals: where popsize is an even integer, Gen is the number of iterations, and Pm is the mutation probability. Second, it deploys a simple and effective local search strategy, double justification, to adjust the chromosomes, as illustrated in the following content, and then calculates fitness for each individual. Third, it uses the crossover operator designed specifically for forbidden time windows (as explained in Section 3) to generate two new offspring, and

mutates them. Thus, the population is enlarged to a size of $2 \cdot \text{popsize}$. Finally, the algorithm sorts the chromosomes and selects the best ones, such that a new population is generated with a size of popsize . This procedure is repeated until Gen is reached or the prescribed CPU time is up.

(1) Initial population and fitness function

The proposed algorithm deploys a precedence feasible activity list to represent activity priorities. This activity list can be used to generate a possible solution using serial SGS. Differing from Hartmann's method, SGS here must take forbidden time windows into account. Some of the solutions in the initial population are randomly assigned, and others are selected from the results of a classic priority-based heuristic algorithm, as mentioned earlier.

Fitness function chooses the highest fitness; thus, the minimum optimized goal has to be transformed to fitness function as

$$f(i) = F_{\max} - S_i$$

where F_{\max} represents the maximum duration of individual i in the latest five generations and S_i the solution resulting from the decoding individual i , that is, the optimized goal.

(2) Double justification

Many scholars have appropriated local search operators in genetic approaches and obtained better schedules. We borrowed a double justification operator to improve the quality of the final results. Thanks to its simplicity, fast speed, and effectiveness, it is convenient to be applied to RCPSP [26,27].

Definition 1. *An active (or left active) schedule is a solution where no activity can be executed in advance without delaying any other activities or violating any rules. Similarly, a right active schedule is one where no activity can be postponed without delaying any other activities, violating any rules, or extending the total makespan.*

Definition 2. *A justification procedure refers to that given a schedule S , after the right (left) justification of an activity $j \neq J(1)$, S becomes a new schedule of S' where $s'_i = s_i, i \neq j; s'_j \geq s_j$ ($s'_j \leq s_j$), and s'_j is as large/small as possible. The procedure is to start activity j as late (early) as possible while the other activities' starting times remain unchanged.*

The double justification of a schedule follows a specific agenda. First, all the activities are sorted in a descending order of their ending times. Second, each activity is justified to the right and the latest completing time of each activity is obtained. The latest completing time of an activity refers to the point that is right before the earliest starting time of its immediate successors and the latest of the time intervals that feed all the recourse constraints. Third, each activity is justified to the left, and all the activities are sorted in an ascending order of their starting times. Fourth, each activity is arranged on its earliest starting time point, such that precedence and recourse constraints are both satisfied. When the procedure is completed, a solution that is no worse than the original one is obtained.

(3) Crossover operators for forbidden time windows

For the proposed time-bound RCPSP model, schedules that accommodate vacant forbidden time windows, meaning no activity is undertaken during those time intervals, must be less efficient than those where all the forbidden time windows are filled with activities. Therefore, to obtain a minimized makespan, we design a crossover operator for forbidden time windows, inspired by the peak crossover operator in Valls et al.'s [18] algorithm. Given a schedule S , $[U_1, U_2]$ represents a forbidden time window, and $SA(u)$ the set of activities being executed during $[U_1, U_2]$, such that $SA(u) = \{i \in J^+; [U_1, U_2] \cap [s_i, c_i] \neq \emptyset\}$. Thus,

the resource utilization ratio (RUR) of schedule S during the special time interval $[U_1, U_2]$ can be described as:

$$RUR(u) = \frac{1}{K} * \sum_{j \in SA(u)} \sum_{k=1}^K \frac{r_{j,k}}{R_k}, \quad 0 \leq RUR(u) \leq 1$$

If $RUR(u)$ is higher than a threshold δ , say 0.7, we can claim that schedule S holds a high RUR during a certain time interval. Let λ be the list of activities that could be executed during a certain special time interval, such that $\lambda = (j_p, j_{p+1}, \dots, j_q)$, and λ could be vacant.

Then, the crossover operator for the forbidden time windows can be used in the following matter. Suppose there are two individuals F, M . If RURs of F and M are both lower than δ , one-point crossover is employed to obtain their off-springs; if either RUR of F or M is higher than δ , two-point crossover is deployed and the two switch points are the starting time and the ending time of each individual, as explained in Figure 4.

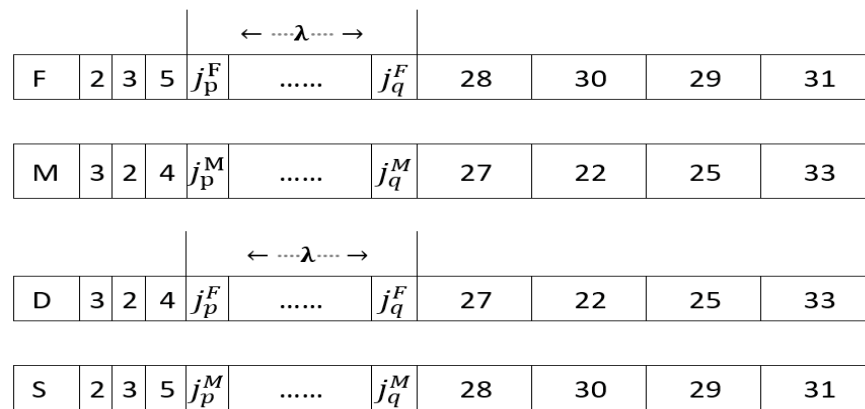


Figure 4. Forbidden time windows crossover operator.

It seems that the crossover operator for forbidden time windows is highly related to activities' RURs during special time intervals. If the RUR of the parent is high, the two-point operator can ensure that this high RUR is passed on to its offspring; if RURs of the parents are below the threshold, the one-point operator can facilitate diversity among the offspring.

(4) Mutation and selection

We applied exchange mutation to the precedence feasible activity list, that is, we randomly selected two genes and switched them according to a mutation probability; if the mutation violated precedence constraints, the two genes were restored back. We also incorporated elite strategy in the selection to guarantee that the best chromosomes would be directly replicated in the next generation. Here the probability of selecting elites is defined as

$$p_i = f(i) / \sum_{j=1}^{popsize} f(j)$$

4. Simulation

We conducted simulated analysis on our proposed algorithm. The simulation was performed on a computer with Intel dule-core CPU at 3.4 GHz. The algorithm was coded in Java.

4.1. Problem Instance Generation

Classic RCPSPs usually use PSPLIB [28], a project scheduling problem library, to test their performance. The problem that this paper targets, however, can hardly make use of the library, due to its extra time constraints. Therefore, we extended the library by adding

special time constraints to those problems. We defined that special time intervals could last between 15~20% of the shortest makespans of the projects recorded in PSPLIB. The starting points of those intervals were randomly assigned along the timelines of the projects, excluding project starting and ending points. We also randomly designated 8~25% of all the activities, excluding dummy ones, as special activities without repetition. Finally, we selected four types of projects which were named J30, J60, J90 and J120, containing four different numbers of activities: 30, 60, 90 and 120; the corresponding numbers of their instances were 480, 480, 480 and 600.

The parameters used in the hybrid algorithm were set as follows: P_m was 0.05; $popsiz$ equals the number of activities of the project; the algorithm would be terminated if Gen reached 200 or the result had not changed in the last 50 generations.

4.2. Results and Comparisons

First, we applied three different methods to one of the instances of J120: a method integrating the parallel scheduling scheme and WCS priority rule, a method integrating serial scheduling scheme and LST priority rule, and the proposed hybrid genetic algorithm. This chosen case included a “forbidden window” lasting a long period and starting at an early stage of the project, and special activities accounting for 12% of the total. The resultant makespans of the three were 209, 288 and 186. The two heuristic algorithms could only determine simple schedules and the quality of the resulted values fluctuated randomly; the proposed algorithm, however, could explore the space of solutions and keep evolving, thus approaching the optimized goals.

Next, we separately combined the parallel scheduling scheme and the serial scheduling scheme with the seven different priority rules (the combinations are mentioned in Section 3.1) and applied them to all the instances of the four types: J30, 60, J90, and J120. Table 3 lists results of the seven PSS-based methods and Table 4 those of the SSS-based methods. The two tables record the number of times that the best solution of each method appears and its average CPU time. Tables 5 and 6 show the average makespans of the PSS-based methods and the SSS-based methods, respectively. Table 7 lists the results of the proposed algorithm.

Table 3. Results of the parallel schedule scheme-based methods.

| | | Priority Rules | | | | | | |
|------|------------------------------------|----------------|-------|-------|------------|-------|-------|-------|
| | | WCS | CA | SPT | LST | MSL | TRD | MTS |
| J30 | The times the best solution appear | 353 | 174 | 194 | 351 | 287 | 181 | 253 |
| | Average CPU time | 9.11 | 10.01 | 9.82 | 9.59 | 9.51 | 9.27 | 9.49 |
| J60 | The times the best solution appear | 339 | 162 | 161 | 346 | 278 | 167 | 206 |
| | Average CPU time/(ms) | 15.73 | 15.96 | 16.36 | 16.75 | 17.4 | 17.19 | 16.95 |
| J90 | The times the best solution appear | 334 | 133 | 149 | 353 | 269 | 145 | 185 |
| | Average CPU time/(ms) | 20.37 | 20.36 | 20.43 | 21.19 | 20.75 | 20.74 | 20.72 |
| J120 | The times the best solution appear | 280 | 53 | 45 | 341 | 107 | 30 | 84 |
| | Average CPU time/(ms) | 31.9 | 31.48 | 31.82 | 31.77 | 31.97 | 32.13 | 32.67 |

Table 4. Results of the serial schedule scheme-based methods.

| | | Priority Rules | | | | | | |
|------|------------------------------------|----------------|-------|-------|------------|-------|-------|------------|
| | | MTS | CA | SPT | LST | MSL | TRD | LFT |
| J30 | The times the best solution appear | 87 | 91 | 3 | 247 | 2 | 8 | 189 |
| | Average CPU time/(ms) | 7.11 | 6.82 | 6.81 | 6.74 | 7.03 | 7.46 | 7.59 |
| J60 | The times the best solution appear | 24 | 59 | 0 | 314 | 0 | 0 | 185 |
| | Average CPU time/(ms) | 15.4 | 15.38 | 16.51 | 16.83 | 16.9 | 17.2 | 17.02 |
| J90 | The times the best solution appear | 20 | 47 | 0 | 348 | 1 | 0 | 168 |
| | Average CPU time/(ms) | 21.91 | 22.56 | 22.82 | 23.14 | 23.33 | 23.28 | 23.36 |
| J120 | The times the best solution appear | 8 | 36 | 0 | 394 | 0 | 0 | 177 |
| | Average CPU time/(ms) | 29.12 | 30.43 | 30.72 | 30.55 | 31.23 | 31.16 | 31.12 |

Table 5. Mean makespans of the parallel schedule scheme-based methods.

| Jobs | WCS | CA | SPT | LST | MSL | TRD | MTS | The Average of the Best Solution |
|------|-----|-----|-----|-----|-----|-----|-----|----------------------------------|
| J30 | 67 | 69 | 70 | 67 | 68 | 71 | 68 | 65 |
| J60 | 91 | 96 | 96 | 91 | 94 | 98 | 94 | 90 |
| J90 | 108 | 115 | 116 | 108 | 111 | 118 | 112 | 107 |
| J120 | 145 | 155 | 159 | 144 | 153 | 167 | 153 | 142 |

Table 6. Mean makespans of the serial schedule scheme-based methods.

| Jobs | MTS | CA | SPT | LST | MSL | TRD | LFT | The Average of the Best Solution |
|------|-----|-----|-----|-----|-----|-----|-----|----------------------------------|
| J30 | 79 | 76 | 91 | 72 | 90 | 89 | 72 | 69 |
| J60 | 115 | 110 | 142 | 100 | 140 | 139 | 102 | 98 |
| J90 | 141 | 133 | 181 | 120 | 175 | 177 | 122 | 118 |
| J120 | 214 | 202 | 280 | 183 | 276 | 275 | 186 | 180 |

Table 7. The genetic algorithm mean makespan.

| Jobs | Makespan | Time (s) |
|------|----------|----------|
| J30 | 65 | 5.12 |
| J60 | 89 | 16.19 |
| J90 | 98 | 25.63 |
| J120 | 126 | 50.6 |

The bolded numbers in Tables 3 and 4 represent the best two results of each type. Among PSS-based methods, the results of WCS and LST are the best, and the results of LST and LFT stand out among SSS-based methods. Comparing Tables 5 and 6, we can find that PSS-based methods perform better than SSS-based, and results of WCS and LST again outstrip others. Figure 5 displays the comparison between mean makespans of PSS-based methods, SSS-based methods and the proposed hybrid genetic algorithm. The figure indicates that the proposed algorithm is not much superior when the projects contain a small number of activities, but as the number of activities rises, the proposed algorithm is better than the others, although its time cost also grows.

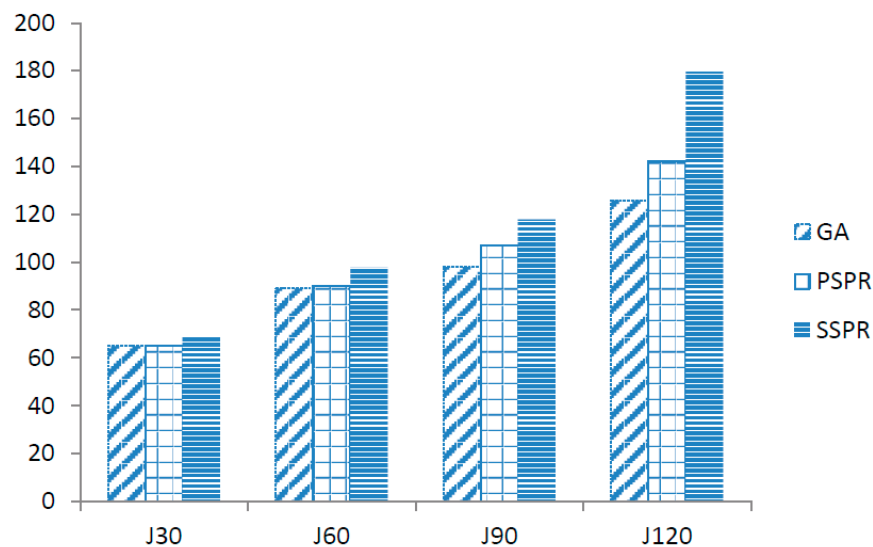


Figure 5. Comparison between the heuristics and the proposed hybrid genetic algorithm.

5. Conclusions

In this paper, we have targeted the water conservancy project scheduling problem. As it is an extension to the classic RCPSP, we have proposed an approach that incorporates a priority rule-based heuristic algorithm and a hybrid genetic algorithm. From the simulation results, we can draw the following conclusions:

- (1) Priority-based heuristic algorithms using the parallel scheduling scheme perform better than those using the serial scheduling scheme;
- (2) Time-constrained priority rules, such as LFT, LST, and WCS can help to obtain better results than other priority rules;
- (3) The proposed algorithm exhibits better performance on large-scale problems, although it costs more time than priority rule-based heuristics.

In this paper, we have presented a forbidden time window water construction project scheduling research that integrates model and solution. This contributes towards building appropriate decision support capabilities in widely diverging project scheduling environments. Given the prevalence of specific time constraints in real life, reducing the impact of time constraints on projects, as discussed in this article, can greatly reduce project duration and cost. However, there are still some issues for further study. For example, more efficient algorithms need to be found, and solutions need to be extended to more practical scenarios.

Author Contributions: Writing—original draft preparation, S.Z.; writing—review and editing, X.S.; visualization, Validation, L.S.; supervision, L.X.; project administration, L.X.; funding acquisition, S.Z. and X.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Science Foundation of China [72204210], Xuzhou medical university excellent talents scientific research start-up fund [53591420]; Research Projects of Philosophy and Social Sciences in Colleges and Universities of Jiangsu Province [2020SJA1079].

Data Availability Statement: Not applicable.

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

References

1. Odedairo, B.O.; Oladokun, V. Relevance and Applicability of Multi-objective Resource Constrained Project Scheduling Problem: Review Article, *Engineering. Technol. Appl. Sci. Res.* **2011**, *1*, 144–150. [[CrossRef](#)]
2. Brucker, P.; Drexel, A.; Möhring, R.; Neumann, K.; Pesch, E. Resource-constrained project scheduling: Notation, classification, models, and methods. *Eur. J. Oper. Res.* **1999**, *112*, 3–41. [[CrossRef](#)]
3. Hartmann, S.; Briskorn, D. A survey of variants and extensions of the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2010**, *207*, 1–14. [[CrossRef](#)]

4. Kolisch, R.; Padman, R. An integrated survey of deterministic project scheduling. *Omega* **2001**, *29*, 249–272. [[CrossRef](#)]
5. Tamás, K. Project scheduling: A review of recent books. *Oper. Res. Lett.* **2005**, *33*, 105–110.
6. Hartmann, S.; Briskorn, D. An updated survey of variants and extensions of the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2022**, *297*, 1–14. [[CrossRef](#)]
7. Pellerin, R.; Perrier, N.; Berthaut, F. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2020**, *280*, 395–416. [[CrossRef](#)]
8. Neumann, K.; Schwindt, C.; Zimmermann, J. Resource-Constrained Project Scheduling with Time Windows. In *Perspectives in Modern Project Scheduling*; Józefowska, J., Weglarz, J., Eds.; International Series in Operations Research & Management Science 92; Springer: New York, NY, USA, 2006; pp. 375–407.
9. Ismael de Azevedo, G.H.; Pessoa, A.A.; Subramanian, A. A satisfiability and workload-based exact method for the resource constrained project scheduling problem with generalized precedence constraints. *Eur. J. Oper. Res.* **2021**, *289*, 809–824. [[CrossRef](#)]
10. Morin, P.-A.; Artigues, C.; Haït, A.; Kis, T.; Spieksma, F.C. A project scheduling problem with periodically aggregated resource-constraints. *Comput. Oper. Res.* **2022**, *141*, 105688. [[CrossRef](#)]
11. Chen, Y.-L.; Rinks, D.; Tang, K. Critical path in an activity network with time constraints. *Eur. J. Oper. Res.* **1997**, *100*, 122–133. [[CrossRef](#)]
12. Zhan, J. Calendarization of time planning in MPM networks. *ZOR Methods Model. Oper. Res.* **1992**, *36*, 423–438. [[CrossRef](#)]
13. Franck, B.; Neumann, K.; Schwindt, C. Project scheduling with calendars. *OR-Spektrum* **2001**, *23*, 325–334. [[CrossRef](#)]
14. Yang, H.-H.; Chen, Y.-L. Finding the critical path in an activity network with time-switch constraints. *Eur. J. Oper. Res.* **2000**, *120*, 603–613. [[CrossRef](#)]
15. Bomsdorf, F.; Derigs, U. A model, heuristic procedure and decision support system for solving the movie shoot scheduling problem. *OR Spectr.* **2008**, *30*, 751–772. [[CrossRef](#)]
16. Lorenzoni, L.L.; Ahonen, H.; de Alvarenga, A.G. A multi-mode resource-constrained scheduling problem in the context of port operations. *Comput. Ind. Eng.* **2006**, *50*, 55–65. [[CrossRef](#)]
17. Vanhoucke, M. Scheduling an R&D project with quality-dependent time slots. In *Computational Science and Its Applications-ICCSA 2006*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 621–630.
18. Mi, X.; Mei, M.; Zheng, X. Study on Optimal Routes of Multimodal Transport under Time Window Constraints. In Proceedings of the IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD), Porto, Portugal, 6–8 May 2019.
19. Drexl, A.; Nissen, R.; Patterson, J.H.; Salewski, F. ProGen/ πx —An instance generator for resource-constrained project scheduling problems with partially renewable resources and further extensions. *Eur. J. Oper. Res.* **2000**, *125*, 59–72. [[CrossRef](#)]
20. Nachtmann, H.; Mitchell, K.N.; Rainwater, C.E.; Gedik, R.; Pohl, E.A. Optimal Dredge Fleet Scheduling Within Environmental Work Windows. *Transp. Res. Rec.* **2014**, *2426*, 11–19. [[CrossRef](#)]
21. Luebsen, J.; Wolken-Moehlmann, G. Comparison of Weather Window Statistics and Time Series Based Methods Considering Risk Measures. In Proceedings of the 17th EERA Deep Sea Offshore Wind R and D Conference (EERA DeepWind), Trondheim, Norway, 15–17 January 2020; IOP Publishing: Bristol, UK, 2020.
22. Naumets, S.; Lu, M.; Ali, M. Project Schedule Development under Environment-Induced Time-Window Constraints: Case of Constructing River-Crossing Bridge in Remote Northern Region. *J. Constr. Eng. Manag.* **2022**, *148*, 04022129.
23. Blazewicz, J.; Lenstra, J.; Kan, A. Scheduling subject to resource constraints: Classification and complexity. *Discret. Appl. Math.* **1983**, *5*, 11–24. [[CrossRef](#)]
24. Kolisch, R. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *Eur. J. Oper. Res.* **1996**, *90*, 320–333. [[CrossRef](#)]
25. Hartmann, S. A competitive genetic algorithm for resource-constrained project scheduling. *Nav. Res. Logist.* **1998**, *45*, 733–750. [[CrossRef](#)]
26. Valls, V.; Ballestín, F.; Quintanilla, S. A hybrid genetic algorithm for the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2008**, *185*, 495–508. [[CrossRef](#)]
27. Valls, V.; Ballestín, F.; Quintanilla, S. Justification and RCPSP: A technique that pays. *Eur. J. Oper. Res.* **2005**, *165*, 375–386. [[CrossRef](#)]
28. Kolisch, R.; Sprecher, A. PSPLIB—A project scheduling problem library: OR Software—ORSEP Operations Research Software Exchange Program. *Eur. J. Oper. Res.* **1997**, *96*, 205–216. [[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.