

Article

Efficient Alternative Mixed-Integer Non-Linear Programs and a Customized Genetic-Based Hybrid Metaheuristic for a Resource-Constrained Project-Scheduling Problem with a Flexible Network

Arash Pourrezaee ¹, Ali Afzali ² and Shahryar Sorooshian ^{3,4,*}¹ Department of Industrial Engineering and Management Systems, Amirkabir University of Technology, Tehran 1591634311, Iran; arash.po@aut.ac.ir² Department of Civil Engineering, Sharif University of Technology, Tehran 14667-54167, Iran; ali_afzali_iut@yahoo.com³ Department of Business Administration, University of Gothenburg, 411 24 Gothenburg, Sweden⁴ Faculty of Engineering and Sustainable Development, University of Gävle, 801 76 Gävle, Sweden

* Correspondence: shahryar.sorooshian@gu.se

Abstract

This paper aims to present efficient alternative mixed-integer non-linear programming models and a customized hybrid metaheuristic, the Genetic-Based Algorithm (GBA), for a resource-constrained project-scheduling problem with a flexible network structure (RCPSP-FNS). We also consider the cost–time trade-off in the problem with a flexible network by using activity-duration compression. We present three approaches to solve the problem, including a mixed-integer non-linear program (MINLP) using binary variables representing activity completion times (MINLP1), an alternative mixed-integer non-linear program using integer variables representing activity-completion times (MINLP2) that has not presented before in RCPSP-FNS modeling, and the GBA. A total of 35 different problems are solved to examine the computational efficiency of the solution approaches. The MINLP1 and MINLP2 models are both solved by the GEKKO solver. The results indicate that solving the MINLP2 model can reach the optimal objective value obtained by solving the MINLP1 model in significantly less time. In addition, the proposed genetic-based algorithm can solve some large problems in a more efficient way in comparison to solving MINLP1 by using GEKKO. However, solving the MINLP2 model using GEKKO is the most efficient solution approach in comparison to both MINLP1 and the proposed genetic-based algorithm. MINLP2 can be solved to proven optimality (in much less time) for problems in which the MINLP1 model can, at most, reach near-optimal solutions.

Keywords: resource-constrained project-scheduling problem (RCPSP); flexible network; mixed-integer non-linear programming (MINLP); genetic algorithm; metaheuristic

MSC: 90C11; 90C27; 90C30; 90C35; 90C59; 90B35

Academic Editor: Ioannis Tsoulos

Received: 6 May 2026

Revised: 27 July 2026

Accepted: 2 August 2026

Published: 5 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

One of the most important aspects of project management is scheduling, which entails determining the best sequence and duration for each project activity [1,2]. An efficient

schedule is required to optimize resource utilization, reduce project duration, and avoid delays [3]. When it comes to managing various tasks with limited resources, specific project delivery periods, and changing project settings, project managers frequently face complex challenges. As a result, it is necessary to use efficient scheduling approaches that can account for the complex nature of projects and thereby optimize resource allocation. The present study addresses the resource-constrained project-scheduling problem with flexible network structures (RCPSP-FNS), a scenario in which project activities can take multiple paths while resources are limited.

This paper introduces an advanced model, and the goal is to increase the computational efficiency to complete these scheduling tasks in terms of both cost and time units. Subsequently, it strengthens project managers' abilities to distribute resources optimally, shorten project durations, and adapt to altering project requirements, all at once. In this paper, we address schedule flexibility in project management by developing a Mixed-Integer Non-Linear Programming (MINLP) model for the Resource-Constrained Project Scheduling Problem with a flexible network structure (RCPSP-FNS) based on the model proposed by Tao and Dong [4], incorporating schedule flexibility. We introduce the MINLP1 model to consider both the cost–time trade-off and schedule flexibility in RCPSP-FNS. Given the NP-hard nature of RCPSPs, we propose a customized Genetic-Based Algorithm (GBA) for solving large problems. Additionally, we contribute to the mathematical formulation by presenting an alternative MINLP2 model, which uses integer variables instead of binary ones, improving efficiency when solved with the GEKKO solver. All models account for both renewable and non-renewable resource constraints and cost–time trade-offs through activity-duration compression.

Following this Introduction, the remainder of this paper is organized as follows: Section 2 reviews the literature to gain a general understanding of how to achieve the research objective; it also compares this research to previous efforts. Section 3 presents the problem description and alternative model formulations. An illustrative example solved by all three solution approaches is presented in Section 4, along with evaluation of the computational efficiency of the alternative solution approaches. Finally, Section 5 presents conclusions and future research directions.

2. Literature Review

In project-scheduling problems, it is necessary to successfully complete a project under different project constraints [5,6]. The resources needed to implement a project are among the most important constraints. In practice, two types of resources can be considered in projects—renewable and non-renewable resources, both of which can be limited [7]. One way to solve the problem of constrained renewable resources is to schedule the project so that the activities have the least overlap with each other to reduce the need for the simultaneous use of resources, which increases the completion time of the project and costs such as tardiness costs [8]. Therefore, it is necessary to use optimization techniques to minimize project duration and costs while maintaining project resource constraints. A large volume of recent research has been conducted on the achievement of the best activity schedule to minimize project completion time (or cost) by considering resource constraints. For example, Li and Zhang [9] presented a model for a Multi-Mode Resource-Constrained Project-Scheduling Problem (MRCPSP) considering renewable and non-renewable resources and solved the model by using the ant colony algorithm. Kyriakidis et al. [10] attempted to construct a new integer formulation for the MRCPSP and Resource-Constrained Project-Scheduling Problem (RCPSP) using renewable and non-renewable sources. They evaluated their formulation by solving several sample problems. Wang and Feng [11] used the Estimation of Distribution Algorithm (EDA) to solve the MRCPSP and used experimental design to select the algorithm parameters. Zamani [12] used a genetic

algorithm to solve the project-scheduling problem with resource constraints and compared the results with other RCPSP solution methods. Naber and Kolisch [13] proposed several integer programming models for the project-scheduling problem with resource constraints considering non-constant resource consumption. Wang and Feng [14] solved the project-planning problem with resource constraints using a hybrid EDA and compared the results with the results of other algorithms.

Paraskevopoulos et al. [15] introduced an innovative event-list-based method to solve the RCPSP and reported much more efficient results compared to other similar methods.

In order to solve the RCPSP, Xiao et al. [16] combined the electromagnetic method with three multi-objective metaheuristic optimization methods including Non-Dominated Sorting Genetic Algorithm-II (NSGA-II), Strength Pareto Evolutionary Algorithm 2 (SPEA2), and Multi-objective Evolutionary Algorithm/Based on Decomposition (MOEA/D). By evaluating their results, they revealed the efficiency of combining the electromagnetic method with the three methods NSGAII, SPEA2 and MOEA/D in improving the efficiency of these methods. Kreter et al. [17] extended the RCPSP by adding a calendar holiday to the model.

In the classic RCPSP model, it is assumed that the project network structure and resources are already known and all activities must be performed, with no flexibility or change in project network structure. In practice, project activities can sometimes be alternatives to each other, meaning that more than one project completion path may be available [4]. Several researchers have studied the RCPSP with a Flexible Network Structure (RCPSP-FNS). For example, Kellenbrink and Helber [18] argue that some airline projects include flexible network structures. Tao and Dong [4] stated that construction projects can normally be executed in different ways, among which just one method is selected in practice. For example, in a bridge construction project, a pier shaft can be constructed in two ways: by casting on site or precasting. Each of these methods includes specific activities and a precedence relationship [19]. The existence of alternative project execution paths is normally modeled using logical concepts like AND/OR. Stochastic project networks were presented by Neumann [20], who showed that activities could only be selected to be implemented while the project was progress. A conditional planning approach was proposed by Tsamardinos et al. [21], in which the selection of an activity in the optimal solution depends on the situation. Tsamardinos et al. [21] also used the temporal constraint satisfaction problem and did not consider all existing project networks. A new concept called the probability of existence (PEX) was proposed by Beck and Fox [22] to model a flexible project structure. The value of PEX was used as a measure for the probability of the existence of an activity in the final solution. Ranjbar and Davari [23] proposed a model to deal with the project-scheduling problem with alternative technologies. They considered research and development projects where the final product may be produced using alternative technologies, at which point just one of the alternative technologies is selected in practice. Several other researches have considered logical constraints for certain classes of projects with flexible network structures. For example, Belhe and Kusiak [24] suggested that design operations may include different logical relationships, like AND, OR and exclusive OR between activities. Gillies and Liu [25] assessed the complexity of RCPSPs with AND/OR project networks and tried to solve the model using heuristic procedures. Čapek et al. [26] considered the resource-constrained project-scheduling problem with different project plans corresponding to different activities, relationships and resource usage.

Tao and Dong [4] proposed a mathematical optimization model (a Mixed-Integer Program (MIP)) to deal with the multi-mode resource-constrained project-scheduling problem (MRCPSP). They proposed a new network representation of project precedence relationships and called it the AND–OR network representation. They also used the

NSGAII and Tabu Search algorithms to solve the problem. One of the most important problems of organizations in project implementation is completing the project in a predetermined time. Due to the competitive conditions prevailing in the market, today the demand for the implementation of projects in the shortest possible time is increasing. These conditions lead to a more critical situation of projects due to the simultaneous impact of resource constraints and uncertainty, which, in turn, plays an important role in the occurrence of delays and interruptions during the project. Activities may exceed the expected time or be completed in less time; resources may be temporarily unavailable, certain activities may be added to the project; or the project may be suspended for a certain period of time, and the processes may change. These are just a few examples of the uncertainties surrounding project implementation. To deal with these uncertainties, the total slack (latest start minus earliest start) of activities' schedules can be increased, which is a preventive approach to project scheduling. By increasing the total slack in a project schedule and adding time buffers to deal with delays, a proper schedule can be created to reduce the likelihood of tardiness.

2.1. Modeling and Solution Methods (Exact and Metaheuristic) for RCPSPs with Flexible Network Structures

Exact solutions to MINLP models of RCPSPs with flexible network structures have been addressed in few studies. For example, Birjandi et al. [27] presented a weighted mixed-integer non-linear model for the RCPSP-FNS (which they call RCPSP with multi-route work packages), where the weights are used to convert the original multiple-objective model to a single-objective MINLP. They just considered the Finish-to-Start (FS) precedence relation in their model. They also used continuous decision variables for activity finish and start times. In addition, they did not consider project-schedule flexibility. They also just considered the cost–time trade-off by using activity execution mode selection (not activity compression). However, due to the computational complexity of the problem, they also used a four-part non-distinct (FPND) solution approach, as well as binary PSO and the GA to solve large problems.

Cajzek and Klanšek [28] presented a novel model for optimizing project schedules considering resource constraints and Alternative Production Processes (APPs). They utilized mixed-integer non-linear programming and offered precise project management outputs. The model's uniqueness lies in its ability to simultaneously optimize production processes and resource-constrained project scheduling within a unified framework. They used continuous variables for activity finish and start times and considered four types of precedence relations (FS as well as Start to Start (SS), Start to Finish (SF), and Finish to Finish (FF)). Like Birjandi et al. [27], Cajzek and Klanšek [28] directly considered neither the cost–time trade-off using activity compression nor project schedule flexibility.

Van der Beeka et al. [29] introduced a method for solving the RCPSP with resource consumption and production while allowing for a flexible network. Recognizing the challenge in selecting an optimal set of activities, the authors proposed a solution based on the concept of "group graphs", specifically targeting projects with non-circular dependency graphs. They presented a hybrid differential evolution algorithm designed to efficiently select and schedule activities, demonstrating its superior performance over traditional algorithms that do not account for resource dynamics. Additionally, they explored the effectiveness of an ant colony optimization algorithm for specific cases, ultimately showing that the hybrid algorithm offers the best solution for a broad range of scenarios. They also used binary variables as activity-scheduling decision variables. However, Van der Beeka et al. [29] considered neither the cost–time trade-off nor schedule flexibility. In addition, they used binary decision variables for activity selection and scheduling. Finally, they considered only FS precedence relations.

Servranckx et al. [30] explored a GA-SAT approach to solve the RCPSP with a flexible network structure. The proposed algorithm was applied to an RCPSP with binary decision variables for scheduling. The model they considered does not include the cost–time trade-off, scheduling flexibility or precedence relations, except FS precedence.

2.2. Modeling and Solution Methods (Exact and Metaheuristic) for Other Variants of the RCPSP

Despite the limited availability of exact solution methods proposed in the literature for the RCPSP with a flexible network, numerous exact solution methods have been suggested for different variants of RCPSPs. For example, the primary focus of Erenguc et al. [31] was addressing the non-preemptive resource-constrained project-scheduling problem with multiple modes. They proposed a branch-and-bound procedure to solve this problem, and their findings are substantiated by computational results involving 160 problems. Their method demonstrates effectiveness in minimizing total project cost, showcasing the practical applicability of their approach. The problem is formulated as an MINLP model without considering a flexible project network or project schedule flexibility. The FS precedence relation is the only type of precedence considered in this paper. In addition, no activity compression (as a method of cost–time trade-off) is modeled. However, they used continuous variables for activity start times.

Atli [32] took on resource-constrained project scheduling problems with deterministic activity durations. The study explored the potential synergy of combining the Tabu Search Algorithm with the CPLEX exact algorithm for management of medium-scale RCPSPs. Atli [32] considered neither the cost–time trade-off, project network flexibility, nor project schedule flexibility. In addition, they considered just FS precedence relations and binary variables for scheduling.

The work by Bruni et al. [33] addresses the robust resource-constrained project scheduling problem, introducing interval uncertainty. They proposed two exact decomposition approaches to tackle this problem, conducting extensive computational experiments to evaluate and compare their methods against state-of-the-art exact solution approaches. They modeled the problem as a two-stage optimization in which the first-stage problem is used for resource allocation and the second-stage problem models project scheduling. In the second-stage problem, there is no consideration of the cost–time trade-off, schedule flexibility, flexible network structure, or precedence relations other than the FS-type relation. In addition, they used continuous variables for activity scheduling.

De Azevedo et al. [34] introduced an exact method to solve the resource-constrained project-scheduling problem with generalized precedence constraints (RCPSP/Max) by transforming a relaxed version into a satisfiability problem. They presented various satisfiability problem formulations and workload-based procedures to narrow down decision-variable domains. They considered neither flexible network structure nor the cost–time trade-off. In addition, schedule flexibility was not considered by De Azevedo et al. [34]. However, different types of precedence relations are supported in their model. They used continuous variables for project scheduling.

Ramos et al. [35] introduced a multi-start iterated local search (MS-ILS) algorithm as a novel solution to the multi-mode resource-constrained project-scheduling problem (MRCPSP), achieving superior outcomes on large instances (100 activities) and competitive results on smaller ones against existing benchmarks. The MRCPSPs studied by Ramos et al. [35] consider neither flexibility in project structure and scheduling nor the cost–time trade-off through activity time compression. They used binary variables for project scheduling.

Shuvo et al. [36] proposed a hybrid metaheuristic approach, integrating Chemical Reaction Optimization (CRO) and the Genetic Algorithm (GA) to solve the RCPSP, showing superior performance and efficiency in comparison to existing algorithms like

adaptive particle swarm optimization (A-PSO), the multi-agent optimization algorithm (MAOA), artificial bee colony (ABC), and the GA. The RCPSP was considered in its simplest form without flexibility in project scheduling and structure and with no cost–time trade-off. They just considered FS precedence and used continuous variables for scheduling.

A critical synthesis of the foregoing literature reveals several persistent and interconnected gaps that collectively limit both the theoretical richness and practical relevance of existing RCPSP-FNS models. First, despite the recognized importance of renewable resources in real-world projects—where labor, equipment, and workspace constraints fundamentally drive scheduling complexity—the vast majority of flexible-network studies [28–30] restrict their attention to non-renewable resources, effectively sidestepping the very constraints that make project scheduling difficult in practice. Second, almost all existing formulations treat schedule flexibility as an afterthought or ignore it entirely, leaving projects vulnerable to inevitable disruptions and offering no mechanism to build robustness into the baseline schedule. Third, cost–time trade-offs are almost exclusively modeled through discrete execution modes rather than continuous duration compression, which oversimplifies managerial decisions and fails to capture the gradation of real-world crashing options. Fourth, the field remains fragmented in its choice of decision variables—binary in some studies [4,29,30] and continuous in others [27,28]—with little critical examination of how this choice affects computational performance, solution quality, or practical interpretability. Finally, exact solution methods are frequently abandoned for large instances without a principled understanding of where the computational bottlenecks lie, while metaheuristics are often applied as generic black-box optimizers rather than being structurally tailored to the unique features of flexible-network problems. These limitations are not merely technical; they reflect a broader tendency in the literature toward descriptive modeling of isolated features rather than integrative, application-driven problem formulation. Our work directly confronts each of these shortcomings—introducing a unified MINLP framework with integer variables that reduces computational burden, incorporating both renewable and non-renewable resources, embedding schedule flexibility directly into the objective, modeling continuous activity compression for realistic cost–time trade-offs, and developing a structure-aware hybrid metaheuristic that scales effectively to large instances—thereby advancing the field from fragmented description toward integrated, practically grounded optimization.

The primary contribution of this paper is a novel Mixed-Integer Non-Linear Programming (MINLP) formulation for the RCPSP-FNS that performs much better than the prior version [4,29,30]. This transformation yields substantial computational advantages, resulting in fewer decision variables and solution times that are significantly lower than equivalent binary formulations when solved with the GEKKO solver. Beyond this core methodological advance, we extend the state of the art in three dimensions previously unexplored in RCPSP-FNS: (i) we introduce schedule flexibility by maximizing the total float in the objective function, creating robust schedules that absorb delays without compromising project completion; (ii) we model cost–time trade-offs through continuous activity duration compression rather than discrete execution modes, providing greater flexibility and realism in representing managerial decisions; and (iii) we address the computational challenge of large-scale instances through a customized hybrid Genetic-Based Algorithm (GBA) that leverages the problem structure via specialized local search operators, achieving superior performance on large problems compared to the exact solution method.

Our model further distinguishes itself by incorporating both renewable and non-renewable resource constraints, extending the flexible-network RCPSP literature, which has predominantly focused on non-renewable resources [28–30]. Renewable resource constraints—such as labor, equipment, and workspace—are particularly critical in practice,

as they limit simultaneous activity execution and represent a primary driver of scheduling complexity in real-world projects. We formulate two MINLP variants (MINLP1 and MINLP2) to directly compare the performance of our integer formulation against the basic formulation, providing empirical evidence for its superiority. Table 1 systematically positions these contributions against prior works. Collectively, these contributions advance both the theoretical modeling of the RCPSP-FNS and its practical applicability through more efficient exact solution methods and effective metaheuristic approaches for large instances.

Table 1. Contributions of this paper compared to similar publications.

Contribution in Model Formulation	Metaheuristic	Exact Optimization	Scheduling Flexibility	Cost–Time Trade-Off	Resource Types	Precedence Relations	Schedule-Related Decision Variables	Flexible network Structure
-	Tabu Search and NSGA-II	Branch and bound (CPLEX)	Inflexible	Using activity modes	Renewable and Non-Renewable	FS	Binary	Flexible
Considering Continuous variables in RCPSP-FNS	Four-part non-distinct (FPND), BPSO, GA	-	Inflexible	Using activity modes	Renewable and Non-Renewable	FS	Continuous	Flexible
-	-	Branch and bound (CPLEX)	Inflexible	Using activity modes	Non-Renewable	FS	Continuous	Inflexible
-	Tabu Search	Branch and bound (CPLEX)	Inflexible	-	Non-Renewable	FS	Continuous	Inflexible
-	-	Decomposition	Inflexible	-	Non-Renewable	FS	Binary	Inflexible
-	Hybrid differential evolution (based on Tabu Search)	-	Inflexible	-	Non-Renewable	FS	Binary	Flexible
Conversion to satisfiability problem	GA-SAT	SAT	Inflexible	-	Non-Renewable	FS	Binary	Flexible
Generalized Precedence		Branch and Reduce (BR) (BARON)	Inflexible	Using activity modes	Non-Renewable	SS, FS, FF, SF	Continuous	Flexible
Using satisfiability	-	Relaxation/Satisfiability	Inflexible	-	Non-Renewable	SS, FS, FF, SF	Continuous	Inflexible
-	Multi-start iterated local search (MS-ILS)	-	Inflexible	Using activity modes	Renewable and Non-Renewable	FS	Binary	Inflexible
-	CRO-GA	-	Inflexible	-	Non-Renewable	FS	Continuous	Inflexible
Novel MINLP model using integer variables	GBA	GEKKO	Flexible	Using activity duration compression	Renewable and Non-Renewable	FS	Integer	Flexible

3. Problem Description and Model Formulation

Consider a project with an activity set (N) that must be executed by utilizing renewable and non-renewable resources (R and U , respectively). If there are precedence relations between project activities, then they must be scheduled according to both precedence and resource constraints. Activity durations are considered to be deterministic, but they can be compressed to be implemented in less time and with more execution cost. The project network structure is flexible, meaning that there is more than one path to complete the project. In other words, project activities must be selected such that they respect precedence and resource constraints while meeting the project objective. The project objective is minimizing total project cost, including fixed costs and compression costs of activities, renewable and non-renewable resource cost, and project-delay cost. We consider an activity-on-node (AON) representation of the project network in which the nodes represent the project activities and the arcs represent precedence relations. We consider the nodes to be of two different types: *AND* nodes and *OR* nodes. All of the successors of *AND* nodes must be selected (as in the traditional RCPSP), whereas just one of the successors of the *OR* nodes needs to be selected). As a strategy to face uncertainties, we model the total float of activities and use it to build a scheduling buffer so that the total float of each activity must be greater than a proportion of the activity duration. We propose the following mixed-integer non-linear program (called MINLP1) for the problem described above.

3.1. MINLP1 Formulation

Sets:

- N : Set of activities indexed with j and i ;
- R : Set of renewable sources indexed by r ;
- U : Set of non-renewable sources indexed by u ;
- T : Set of time periods indexed by t and τ .

Parameters:

- A_{ij} : Adjacency matrix, where $A_{ij} = 1$ it means that node i is a direct predecessor for node j and node j and $A_{ij} = 0$ otherwise;
- P_t : Parameter defining the time period;
- h_i : Node type—for $i \in N \setminus \{n + 1\}$, if $h_i = 1$, i is an *AND* node and $h_i = 0$ otherwise;
- D_i : The duration of activity i ;
- T_i^{max} : Maximum compression time for activity i ;
- C_i : The operating cost of activity i (excluding the cost of resources);
- Re_{ir} : The amount of renewable resources (r) required when activity i is executed;
- NRe_{iu} : The amount of non-renewable resources (u) required when activity i is executed;
- $\overline{R_r}$: The total amount of renewable resources (r) allocated to the project;
- $\overline{U_u}$: The total amount of non-renewable resources (u) allocated to the project;
- $T^{deadline}$: The latest time allowed for project completion;
- C^p : The penalty cost per unit of time in case of delay;
- cr_i : The cost of reducing one unit of duration of activity i ;
- Rc_r : Unit cost of renewable resource r ;
- Uc_u : Unit cost of non-renewable resource u ;
- δ : Minimum fraction of duration of each activity to be considered as the total float;
- M : A sufficiently large number.

Decision Variables

T^{delay} : The project completion delay;

x_i : $x_i = 1$ if node i is selected and $x_i = 0$ otherwise;

y_{it} : $y_{it} = 1$ if activity i is completed at the end of time period t and $y_{it} = 0$ otherwise;

α_{it} : $\alpha_{it} = 1$ if activity i is compressed by t units of time and $\alpha_{it} = 0$ otherwise;

TS_i : Total float of activity i ;

es_i : The earliest start time for activity i ;

ls_i : The latest start time of activity i ;

Qes_{ij} : Slack variable defining the relationship between the earliest starts time for activities i and j ;

Qls_{ij} : Slack variable defining the relationship between the latest start times of activities i and j .

Mathematical Model

$$\min z = \sum_{i \in N} x_i * [c_i + \sum_{t \in T} cr_i * P_t * \alpha_{it} + \sum_{r \in R} Rc_r Re_{ir} + \sum_{u \in U} Uc_u NRe_{iu}] + C^P * T^{delay} \quad (1)$$

S.t:

$$x_1 = 1 \quad (2)$$

$$A_{ij} * x_i \geq x_j \quad \forall i, j \in N \mid h_i = 1 \quad (3)$$

$$\sum_j A_{ij} * x_j \leq x_i \quad \forall i \in N \mid h_i = 0 \quad (4)$$

$$\sum_{t \in T} P_t * y_{it} - (2 - (x_i + x_j)) * M \leq \sum_{t \in T} P_t * y_{jt} - D_j + \sum_{t=1}^{T_i^{max}} P_t * \alpha_{it} \quad \forall i, j \in N \mid A_{ij} = 1 \quad (5)$$

$$\sum_{t \in T} y_{it} = x_i \quad \forall i \in N \quad (6)$$

$$\sum_{i \in N} \left(\sum_{t=P_t}^{P_t + (D_i - P_t * \alpha_{it}) - 1} (Re_{ir} * y_{it}) \right) \leq \bar{R}_r \quad \forall r \in R, t \in T \quad (7)$$

$$\sum_{i \in N, t \in T} NRe_{iu} * y_{it} \leq \bar{U}_u \quad \forall u \in U \quad (8)$$

$$T^{delay} \geq \sum_{t \in T} P_t * y_{|N|t} - T^{deadline} \quad (9)$$

$$T^{delay} \geq 0 \quad (10)$$

$$es_j = \sum_{i \mid A_{ij}=1} Qes_{ij} * x_i * \left(es_i + D_i - \sum_{t=1}^{T_i^{max}} P_t * \alpha_{it} \right) \quad \forall j \in N \quad (11)$$

$$es_1 = 0 \quad (12)$$

$$\sum_i Qes_{ij} \leq 1 \quad \forall j \in N \quad (13)$$

$$ls_j = \sum_{i|A_{ji}=1} Qls_{ji} * x_i * \left(ls_i - D_j + \sum_{t=1}^{T_i^{max}} P_t * \alpha_{jt} \right) \quad \forall j \in N \quad (14)$$

$$ls_{|N|} \leq |T| \quad (15)$$

$$\sum_i Qls_{ji} \leq 1 \quad \forall j \in N \quad (16)$$

$$TS_j = ls_j - es_j \quad \forall j \in N \quad (17)$$

$$TS_j \geq \delta * D_j * x_j \quad \forall j \in N \quad (18)$$

$$\sum_{t \in T} \alpha_{jt} \leq x_j \quad \forall j \in N \quad (19)$$

$$x_i, y_{it}, \alpha_{it}, Qls_{ji}, Qes_{ji} \in \{0,1\}, \forall i, j, t \quad (20)$$

$$ls_j, es_j, TS_j \in Int \quad (21)$$

Equation (1) is the objective function including the total project cost, including the fixed cost and compression cost of activities, renewable and non-renewable resource costs, and project-delay cost.

Equation (2) sets the value for activity number 1 (dummy activity for project start).

Constraints (3) and (4) deal with activity selection for AND nodes and OR nodes, respectively. Note that these inequalities are always valid if there is no precedence relation between two activities.

Precedence relations are considered in Constraint (5). Again, this inequality is effective only when there is a precedence relation between two activities. The left-hand side of this constraint is equal to the finish time of the predecessor activity (i ; if activities i and j are both selected for execution) or a big number (M) (if either activity i or j is not selected for execution). The right-hand side is also equal to the start time of successor activity j , considering nominal duration (D) and duration compression.

Constraint (6) ensures that the activity completion time (y) can take a non-zero value only if the activity is selected for implementation. Constraints (7) and (8) represent resource constraints for renewable and non-renewable resources respectively. Note that renewable resources are engaged only during the execution of related activities assigned to them.

Constraints (9) and (10) model the project delay. According to Constraint (9), the value of project delay must be greater than the length of the time interval between the finish time of the last project activity (which is a dummy activity of project completion) and the project deadline. If T^{delay} is greater than 0, the objective function selects the minimal value of this variable in the optimal solution. If the project completes before the deadline, the value of T^{delay} equals 0.

Constraints (11) to (13) model the earliest start times for project activities. The earliest start time of each successor activity (j) is calculated as the maximum of the earliest start times of its predecessors (Constraints (11) and (12)) with the initial condition of $es_1 = 0$ (Constraint (13)).

Constraints (14) to (16) represent the latest start times for project activities. The latest start time of each predecessor activity (i) is calculated as the minimum of the latest start times of its successors (Constraints (14) and (15)) with the initial condition of $ls_{|N|} \leq |T|$ (Constraint (16)).

The total float is modelled in Equations (17) and (18). Constraint (17) defines the total slack of each activity as the difference between its latest and earliest start times. Constraint (18) forces the model to set a total slack greater than or equal to a certain fraction (δ) of the activity duration.

Allowable activity compression times are modelled in Constraint (20), meaning that activity compression can only be applied to the activities selected for execution. Finally, Constraints (21) and (22) represent the domains of the decision variables. The MINLP1 model includes Constraints (11)–(19) more than the model presented by Tao and Dong [4]. This is because MINLP1 considers activity compression and schedule flexibility in addition to what was modelled by Tao and Dong [4].

3.2. Proposed Customized Hybrid Genetic-Based Algorithm (GBA)

In this section, we propose a customized hybrid genetic-based algorithm (GBA) to solve the RCPSP with a Flexible Network Structure (RCPSP-FNS). Our proposed algorithm uses heuristics to generate initial solutions (instead of random solutions (as occurs in the Genetic Algorithm)). The heuristics are also used in the improvement phase, along with traditional genetic operators (e.g., crossover and mutation). In addition, we do not use matrix representation as solution representation approach (like what occurs in the traditional Genetic Algorithm (GA)). Instead, we use the dictionary data type in our computer programming in Python 3.13.

The GBA differs from the basic Genetic Algorithm from several perspectives:

- The primary uniqueness of the GBA lies in the initial population generation process. The GBA uses a problem-specific construction procedure that guarantees network-structure feasibility, unlike basic genetic algorithms, which typically generate completely random solutions. While conventional GAs initialize populations using purely random binary vectors or permutation-based encodings, our Algorithm 1 employs a topologically aware node-selection process that respects the logical AND/OR relationships in the flexible network. Specifically, the algorithm traverses nodes sequentially and enforces feasibility rules: AND nodes require all predecessors to be selected, while OR nodes select exactly one predecessor via uniform random selection. This ensures that every generated solution represents a valid execution path through the network, dramatically reducing the infeasible solution space that basic GAs would waste time exploring. Additionally, Algorithms 2–4 complement this by generating structured starting times, duration compression and float values that are problem-aware: activities are assigned start times uniformly across the feasible horizon, and compression levels are randomly initialized within individual activity bounds, creating a diverse initial population that already satisfies temporal and resource constraints.
- The crossover operator in the GBA differs fundamentally from standard GA crossover in its multi-component, feasibility-preserving design that operates simultaneously on three heterogeneous decision vectors (network selection (x), start times (S), and compression levels (α)), whereas basic GAs typically crossover a single chromosome encoding. Our single-point crossover (Algorithm 5) performs coordinated recombination at identical cut points across all three parent components, producing offspring that inherit coherent subsets of parental schedules while maintaining the feasibility of the network structure. More critically, the crossover operator incorporates an integrated repair mechanism through post-crossover recalculation: after combining parental vectors, the algorithm recomputes all dependent variables, including durations, finish times, delays, earliest/latest start times, and total slacks using dedicated subroutines (Algorithms 3 and 4). This ensures that every offspring has

consistent and feasible activity timings, unlike basic GAs that typically require separate repair operators or penalty functions for constraint handling.

- The GBA's mutation operator distinguishes itself through its targeted, structure-preserving perturbation strategy compared to the random bit-flipping or Gaussian mutation common in basic GAs. Rather than mutating multiple genes arbitrarily, Algorithm 6 selects a single random activity and applies coordinated mutations across its start time, compression level, and resulting finish time simultaneously, ensuring that the temporal consistency of the entire schedule is maintained. This focused perturbation is fundamentally different from basic GA mutation, which typically modifies individual genes independently without considering the propagation of changes through the schedule network. More importantly, the GBA mutation incorporates immediate feasibility restoration by recalculating all schedule-dependent metrics—including the total project delay and activity total slack (TS)—immediately after mutation, ensuring that the offspring remains a valid schedule. This is a significant departure from basic GAs that rely on iterative repair heuristics or expensive constraint validation. Additionally, the mutation maintains upper-bound awareness by constraining compression mutations within activity-specific maximum limits (0 to $T_k^{\max}$) and start times within the project horizon, preventing the generation of unrealistic schedules, whereas basic GAs often generate infeasible solutions that are either discarded or penalized, reducing search efficiency.

The pseudocodes for all the parts of our proposed algorithm are presented below:

GA Overall Pseudocode

Step 1: Define GA parameters (including npop (number of solutions in the population), nCross (number of crossover operations in each iteration), nMut (number of mutation operations in each iteration), and the algorithm termination condition).

Step 2: Generate initial solutions (initial population) and evaluate the fitness for each solution.

Step 3: If the specified termination condition is not reached, complete the following sub-steps:

Step 3-1: Perform the crossover operation, generate nCross new solutions, calculate their fitness, and add them to the initial population.

Step 3-2: Perform the mutation operation, generate nMut new solutions, calculate their fitness, and add them to the population in Step 3-1.

Step 3-3: Sort the population in ascending order, and keep the top npop solutions as the new population.

Generating initial solutions

In this subsection, we present the algorithms used to generate initial solutions. In this step, Algorithm 1 is used for node selection, as presented and explained below:

1. Set the initial node selection as $x_1 = 1$.
2. Initialize the set of candidate nodes to be considered as $Actset = \{2, 3, \dots, |N|\}$, where $|N|$ is the total number of nodes in the network.
3. While there are still nodes in the $Actset$:
 - a. Select the first node from $Actset$ as j .
 - b. Find all nodes (i) such that there is a connection between node i and node j ($A_{ij} = 1$) and name it set f .
 - c. If the sum of selected nodes (i) from the connections is less than the total number of connections ($|f|$), set $x_j = 0$ and remove node j from $Actset$.
 - d. If condition (c) is not met and node j is an AND node ($h_j = 1$), select all nodes (i) connected to node j and set $x_j = 1$ and $x_i = 1$. Remove node j from $Actset$.

- e. If node j is an OR node and does not meet condition (c), select the node (r) in set f with the $U(f)$ function, which is the uniform selection of one element in f . Set $x_j = 1$ and $x_r = 1$, and remove set f from $Actset$, adding nodes j and r to the $Actset$.
4. Once all nodes have been considered, set the last node in the network as selected ($x_{j|j=|N|} = 1$).

Algorithm 1: Node selection

```

 $x_1 = 1$ 
 $Actset = \{2, 3, \dots, |N|\}$ 
While  $|Actset| > 0$ 
   $j = Actset(1)$ 
   $f = \{i | A_{ij} = 1\}$ 
  if  $\sum_{i \in f} x_i < |f|$ 
     $x_j = 0$ 
     $Actset \leftarrow Actset \setminus j$ 
  else
    if  $h_j = 1$ 
       $f = \{i | A_{ji} = 1\}$ 
       $x_j = 1$ 
       $x_{i|i \in f} = 1$ 
       $Actset \leftarrow Actset \setminus j$ 
    else
       $f = \{i | A_{ji} = 1\}$ 
       $r = U(f)$ 
       $x_j = 1$ 
       $x_r = 1$ 
       $Actset \leftarrow Actset \setminus \{f \setminus r \cup j\}$ 
    end if
  end if
end while
 $x_{j|j=|N|} = 1$ 

```

Algorithm 2 is used for activity duration compression and is explained below:

1. For each node (j) from 1 to the total number of nodes in the network:
2. If node j is the first node or the last node in the network ($j = 1$ or $j = |N|$), set the α_j parameter to 0.
3. If node j is not the first or last node in the network, generate a random value for α_j using a uniform distribution between 0 and the maximum compression of node j (T^{max}_j).

Algorithm 2: Activity duration compression

```

for  $j = 1:|N|$ 
  if  $j = 1$  or  $j = |N|$ 
     $\alpha_j = 0$ 
  else
     $\alpha_j = \text{uniform}(0, T^{max}_j)$ 
  end if
end for

```

Algorithm 3 is used for activity scheduling and works as follows:

1. Identify the nodes where x_j is equal to 0 and store them in set f .
2. Set the starting times (S) and finishing times (F) for the nodes in set f to be 0.

3. Initialize the starting time and finishing time for the first node to be 1.
4. Adjust the active set (*Actset*) to exclude the nodes in set *f*.
5. Calculate the duration for each node in the active set based on $D_j - \alpha_j$.
6. Assign starting times for the nodes in the *Actset* using a uniform distribution between 1 and $T^{deadline}$.
7. Calculate the finishing times for the nodes in the active set based on their starting times and durations.
8. Determine the latest finishing time among the nodes and calculate any delay beyond the deadline.

Algorithm 3: Activity scheduling

```

f = {j | xj = 0}
 $S_{j|j \in f} = 0$ 
 $F_{j|j \in f} = 0$ 
 $S_1 = 1$ 
 $F_1 = 1$ 
 $Actset = N \setminus f$ 
for j ∈ Actset
     $Duration_j = D_j - \alpha_j$ 
     $S_j = uniform(1, T^{deadline})$ 
     $F_j = S_j + Duration_j$ 
end for
 $F_{|N|} = \max_j (F_{j|j < |N|})$ 
 $T^{delay} = \max(0, F_{|N|} - T^{deadline})$ 

```

Algorithm 4 is used to calculate the earliest start time (*es*) and latest start time (*ls*) for each activity in a project schedule. Briefly, the algorithm works as follows:

1. Initialize the earliest start time for all nodes to 0, except for the first node, which starts at time 1.
2. Calculate the earliest start times for activities in the project schedule based on their dependencies and durations.
3. Initialize the latest start time for all nodes to 0, except for the last node, which ends at time *T* (total project duration).
4. Calculate the latest start time for activities in the project schedule based on their dependencies and durations.
5. Calculate the total slack time (*TS*) for each activity by subtracting the earliest start time from the latest start time.

Algorithm 4: Activity float calculation

```

 $es_{j|j \in N} = 0$ 
 $es_1 = 1$ 
 $Actset = N \setminus \{1\}$ 
While |Actset| > 0
    j = Actset1
    if xj = 0
         $es_j = 1$ 
         $Actset \leftarrow Actset \setminus j$ 
    else
         $f = \{i | A_{ij} = 1\} \setminus \{i | x_i = 0\}$ 
        if  $\{i | i \in f, es_i = 0\} = \emptyset$ 
             $es_j = \max_{i \in f} \{es_i + Duration_i\}$ 
             $Actset \leftarrow Actset \setminus j$ 
        end if
    end if
end while

```

```

    end if
end while

 $ls_j |_{j \in N} = 0$ 
 $ls_{|N|} = |T|$ 
 $Actset = N \setminus \{|N|\}$ 
While  $|Actset| > 0$ 
     $j = Actset_{|Actset|}$ 
    if  $x_j = 0$ 
         $ls_j = |T|$ 
         $Actset \leftarrow Actset \setminus j$ 
    else
         $f = \{i | A_{ji} = 1\} \setminus \{i | x_i = 0\}$ 
        if  $\{i | i \in f, ls_i = |T|\} = \emptyset$ 
             $ls_j = \min_{i \in f} \{ls_i - Duration_i\}$ 
             $Actset \leftarrow Actset \setminus j$ 
        end if
    end if
end while

for  $j = 1: |N|$ 
     $TS_j = ls_j - es_j$ 
end for

```

In brief, Algorithm 5 is used for crossover in the Genetic Algorithm-based approach to solving the RCPSP with a flexible network structure. The crossover operation on a sample binary vector like x is illustrated in Figure 1 (as well as the mutation operator on an integer vector like S). The crossover operator involves the following main steps:

1. Select two parent solutions randomly from the population ($i1$ and $i2$).
2. Generate two new solutions by combining parts of the selected parent solutions based on a random crossover point ($Newsol^1$ and $Newsol^2$).
3. Calculate the start and finish times, durations, and delays for the two new solutions.
4. Determine the maximum finishing time and calculate any delay beyond the deadline for each new solution.
5. Calculate additional variables such as the earliest start time (es), latest start time (ls), and total slack time (TS) for the new solutions using Algorithm 4.

Algorithm 5: Crossover

```

 $i1 = \text{uniform}(1, npop)$ 
 $i2 = \text{uniform}(1, npop)$ 
Generate  $x^{i1}$  and  $x^{i2}$  using Algorithm 1
 $Newsol^1.x = x^{i1}$ 
 $Newsol^2.x = x^{i2}$ 
 $k = \text{uniform}(2, |N| - 1)$ 
 $Newsol^1.S = Newsol^1.x * [sol(i1).S[1:k], sol(i2).S[k+1:|N|]]$ 
 $Newsol^2.S = Newsol^2.x * [sol(i2).S[1:k], sol(i1).S[k+1:|N|]]$ 
 $Newsol^1.\alpha = [sol(i1).\alpha[1:k], sol(i2).\alpha[k+1:|N|]]$ 
 $Newsol^2.\alpha = [sol(i2).\alpha[1:k], sol(i1).\alpha[k+1:|N|]]$ 
 $Duration^1 = D - Newsol^1.\alpha$ 
 $Duration^2 = D - Newsol^2.\alpha$ 
 $Newsol^1.F = Newsol^1.S + Duration^1$ 
 $Newsol^2.F = Newsol^2.S + Duration^2$ 
 $Newsol^1.F_{|N|} = \max_j (Newsol^1.F_{j | j < |N|})$ 

```

$Newsol^2.F_{|N|} = \max_j(Newsol^2.F_{j|j < |N|})$
 $Newsol^1.T^{delay} = \max(0, Newsol^1.F_{|N|} - T^{deadline})$
 $Newsol^2.T^{delay} = \max(0, Newsol^2.F_{|N|} - T^{deadline})$
 Calculate the variables es , ls and TS for the solutions $Newsol^1$ and $Newsol^2$ using the Algorithm 4

Algorithm 6 is used for mutation and involves the following steps:

1. Select a solution from the population randomly.
2. Create a new solution by copying the selected solution's parameters.
3. Introduce a random mutation in the start time, duration, and finishing time of a randomly selected activity within the solution.
4. Calculate the necessary variables for the mutated solution, such as finish times, delays, earliest start time (es), latest start time (ls), and total slack time (TS), using Algorithm 4.

Algorithm 6: Mutation

$i = \text{uniform}(1, n_{pop})$
 $Newsol.x = x^i$
 $Newsol.\alpha = sol(i).\alpha$
 $Newsol.S = sol(i).S$
 $Newsol.F = sol(i).F$
 $k = \text{uniform}(2, |N| - 1)$
 $Newsol.S_k = \text{uniform}(1, |T|)$
 $Newsol.\alpha_k = \text{uniform}(0, T^{max}_k)$
 $Duration = D - Newsol.\alpha$
 $Newsol.F_k = Newsol.S_k + Duration_k$
 $Newsol.F_{|N|} = \max_j(Newsol.F_{j|j < |N|})$
 $Newsol.T^{delay} = \max(0, Newsol.F_{|N|} - T^{deadline})$
 Calculate the variables es , ls and TS for the solution $Newsol$ using the Algorithm 4

Sol1	0	1	0	0	1	1	0	Crossover, K=4	NewSol1	1	0	0	1	1	1	0
Sol2	1	0	0	1	1	0	0		NewSol2	0	1	0	0	1	0	0
Sol	1	3	5	2	5	4	5	Mutation, K=3	New Sol	1	3	3	2	5	4	5

Figure 1. Crossover and mutation operators.

3.3. Alternative Mixed-Integer Non-Linear Program (MINLP2)

In this sub-section, we propose a different mathematical model for the resource-constrained project-scheduling problem with a flexible network structure. The proposed model uses binary slack variables to measure the overlap between the project activities, then uses integer variables for sequencing the activities. This model (hereafter referred to as MINLP2) is presented below:

Sets:

- N: Set of activities indexed with j and i ;
- R: Set of renewable sources indexed by r ;
- U: Set of non-renewable sources indexed by u .

Parameters:

A_{ij} : Adjacency matrix, where $A_{ij} = 1$ means that node i is a predecessor for node j and node j and $A_{ij} = 0$ otherwise;

- h_i : Node type, where, for $i \in N \setminus \{n + 1\}$, if $h_i = 1$, i is an *AND* node; and $h_i = 0$ otherwise;
- D_i : The duration of activity i ;
- T^{max}_i : Maximum compression time for activity i ;
- C_i : The operating cost of activity i (excluding the cost of resources);
- Re_{ir} : The amount of renewable resources (r) required when activity i is executed;
- NRe_{iu} : The amount of non-renewable resources (u) required when activity i is executed;
- $\overline{R}_r$: The total amount of renewable resources (r) allocated to the project;
- $\overline{U}_u$: The total amount of non-renewable resources (u) allocated to the project;
- $T^{deadline}$: The latest time allowed for project completion;
- CP : The penalty cost per unit of time in case of delay;
- cr_i : The cost of reducing one unit of duration of activity i ;
- Rc_r : Unit cost of renewable resource r ;
- Uc_u : Unit cost of non-renewable resource u ;
- δ : Minimum fraction of duration of each activity to be considered as the total float;
- M : A sufficiently large number;
- H = Planning horizon;
- A^2_{ij} : Indirect adjacency matrix, where $A^2_{ij} = 1$ means that node i is a direct or indirect predecessor for node j and $A^2_{ij} = 0$ otherwise;
- A^3_{ij} : Symmetric indirect adjacency matrix, where $A^3_{ij} = 1$ means that node i is a direct or indirect predecessor for node j or vice-versa and $A^3_{ij} = 0$ otherwise;
- b_{ij} : Common resource matrix, where $b_{ij} = 1$ means that nodes i and j need common resources and $b_{ij} = 0$ otherwise;
- g_{ij} : Common resource exceeding matrix, where $g_{ij} = 1$ means that nodes i and j need at least one common resource $\otimes$ more than available resource $\overline{R}_r$ in case of an overlapped schedule, and $g_{ij} = 0$ otherwise;
- ss_{ij} : $ss_{ij} = 1$ means that nodes i and j need at least one common resource $\otimes$ more than available resource $\overline{R}_r$ in case of an overlapped schedule while $A^3_{ij} = 0$ and $ss_{ij} = 0$ otherwise.
- ps_{ij} : $ps_{ij} = 1$ means that nodes i and j need at least one common resource while $A^3_{ij} = 0$ and $g_{ij} = 0$ and $ps_{ij} = 0$ otherwise.

Decision Variables

Binary variables

- x_i : $x_i = 1$ if node i is selected and $x_i = 0$ otherwise.
- v^1_{ij} : $v^1_{ij} = 1$ if activity i starts before activity j and $v^1_{ij} = 0$ otherwise.
- v^2_{ij} : $v^2_{ij} = 1$ if activity i finishes after the start of activity j and $v^2_{ij} = 0$ otherwise.

Integer variables

- S_i : Start time of activity i ;
- F_i : Finish time of activity i ;
- α'_i : Compression time for activity i ;
- T^{delay} : The project completion delay;
- TS_i : Total float of activity i ;
- es_i : The earliest start time for activity i ;
- ls_i : The latest start time of activity i ;
- lf_i : The latest finish time of activity i ;
- Qes_{ij} : Slack variable defining the relationship between the earliest start times for activities i and j ;

Qls_{ij} : Slack variable defining the relationship between the latest start times of activities i and j ;

Mathematical Model

$$\min z = \sum_{i \in N} x_i * [c_i + cr_i * \alpha'_i + \sum_{r \in R} Rc_r Re_{ir} + \sum_{u \in U} Uc_u NRe_{iu}] + C^P * T^{delay} \quad (22)$$

S.t:

$$x_1 = 1 \quad (23)$$

$$A_{ij} * x_i \leq x_j \quad \forall i, j \in N \mid h_i = 1 \quad (24)$$

$$\sum_j A_{ij} * x_j \geq x_i \quad \forall i \in N \mid h_i = 0 \quad (25)$$

$$x_i * x_j * F_i \leq S_j + (lf_i - es_j) * v^1_{ji} \quad \forall i, j \in N \mid ss_{ij} = 1, i \neq j \quad (26)$$

$$x_i * x_j * F_i \leq S_j \quad \forall i, j \in N \mid A^2_{ij} = 1 \quad (27)$$

$$v^1_{ij} + v^1_{ji} \leq x_i * x_j \quad \forall i, j \in N \mid A^3_{ij} = 1 \text{ or } ss_{ij} = 1, i \neq j \quad (28)$$

$$x_i * x_j * S_j \leq S_i + (ls_j - es_i) * v^1_{ij} \quad \forall i, j \in N \mid ps_{ij} = 1, i > j \quad (29)$$

$$x_i * x_j * (F_j - S_i) \leq (lf_j - es_i) * v^2_{ji} \quad \forall i, j \in N \mid ps_{ij} = 1, i \neq j \quad (30)$$

$$v^1_{ij} \leq v^2_{ji} \quad \forall i, j \in N \quad (31)$$

$$F_j = x_j * (S_i + D_i - \alpha'_i) \quad \forall j \in N \quad (32)$$

$$S_j \leq x_j * M \quad \forall j \in N \quad (33)$$

$$F_j \leq H \quad \forall j \in N \quad (34)$$

$$x_j * (Re_{jr} + \sum_{i \mid i \neq j, ps_{ji}=1} x_i * Re_{ir} * (v^2_{ij} - v^1_{ji})) \leq \bar{R}_r \quad \forall r \in R, j \in N \quad (35)$$

$$\sum_{j \in N} NRe_{ju} * x_j \leq \bar{U}_u \quad \forall u \in U \quad (36)$$

$$T^{delay} \geq S_{|N|} - T^{deadline} \quad (37)$$

$$T^{delay} \geq 0 \quad (38)$$

$$es_j = \sum_{i \mid A_{ij}=1} Qes_{ij} * x_i * (es_i + D_i - \alpha'_i) \quad \forall j \in N \quad (39)$$

$$es_1 = 0 \quad (40)$$

$$\sum_i Qes_{ij} \leq 1 \quad \forall j \in N \quad (41)$$

$$ls_j = \sum_{i \mid A_{ji}=1} Qls_{ji} * x_i * (ls_j - D_j + \alpha'_j) \quad \forall j \in N \quad (42)$$

$$ls_{|N|} = H \quad (43)$$

$$\sum_t Qls_{ji} \leq 1 \quad \forall j \in N \quad (44)$$

$$lf_j = x_j * (ls_j + D_j - \alpha'_j) \quad \forall j \in N \quad (45)$$

$$TS_j = ls_j - es_j \quad \forall j \in N \quad (46)$$

$$TS_j \geq \delta * D_j * x_j \quad \forall j \in N \quad (47)$$

$$\alpha'_j \leq x_j * T^{max}_j \quad \forall j \in N \quad (48)$$

$$x_i, Qls_{ji}, Qes_{ji} \in \{0,1\}, \forall i, j, t \quad (49)$$

$$ls_j, es_j, TS_j, S_j, F_j, \alpha'_j \in Int \quad \forall j \in N \quad (50)$$

Equation (22) presents the objective function (same as that of MINLP1, the only difference being the definition of activity compression as an integer variable). Equation (23) sets the dummy activity for the project start. Constraints (24) and (25) addresses activity selection. Equations (23)–(25) are identical for MINLP1 and MINLP2. Constraints (26)–(30) represent precedence relations. Constraint (26) ensures that if two activities (*i* and *j*) require more common resources than the available total renewable resources ($ss_{ij} = 1$), these activities are scheduled sequentially.

Constraint (27) ensures that if two activities (*i* and *j*) have precedence relations ($A^2_{ij} = 1$), these activities are scheduled sequentially. Constraint (28) states that if $A^3_{ij} = 1$ or $ss_{ij} = 1$, (at most) one of variables v^1_{ij} and v^1_{ji} can be equal to one. Equations (29)–(31) model the schedule of the pairs of activities, which can overlap (in terms of both precedence and resource constraints). Note here that two activities (*i* and *j*) can overlap only if $v^2_{ij} = 1$ and $v^1_{ji} = 0$.

Constraint (34) assigns the activity finish times. Constraints (33) and (34) represent boundary limits of start and finish times. Constraint (35) represents a renewable resource constraint. Note that each activity's renewable resource usage plus the renewable resources used by its overlapping activities (for which we have $v^2_{ij} - v^1_{ji} \leq 1$) cannot exceed the amount of total available renewable resources of each type. Constraint (36) represents a non-renewable resource constraint.

Constraints (37) and (38) represent the project delay. Constraints (39) to (47) the activity float times similarly to MINLP1. Constraint (48) modes the allowable activity compression times. Finally, variable domains are presented in Constraints (49) and (50).

4. Computational Experiments

After presenting two alternative mixed-integer non-linear programming models and the GBA for the resource-constrained project-scheduling problem with a flexible network structure, in this section, we will examine the computational results obtained by solving several sample problems in different dimensions using each of the presented solution approaches. In addition to the GEKKO and GBA, we also used the Pymoo standard Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) for comparison and validation of the GBA. In this section, an illustrative problem is fully explained in order to validate the model. In addition, twenty problems in different dimensions are solved and reported in order to evaluate the computational efficiency of the presented solution approaches in larger dimensions. The exact solutions for the problems are provided by using GEKKO solver in Python (for MINLP and MINLP2). In addition, the proposed GBA is

implemented in Python. All the calculations are performed on a computer with a Corei5 2.67 GHz CPU and 4 gigabytes of RAM.

The GBA nCross (number of crossover operations in each iteration) and nMut (number of mutation operations in each iteration) parameters are set to be 0.3 and 0.7, respectively, by using the Taguchi method [37]. The npop parameter (number of solutions in the population) is set to 100 for all the metaheuristics.

4.1. Illustrative Example

In order to validate the presented models, a problem with a small size and its optimal solution are described in this sub-section. The sample problem has 12 real activities (activities 2 to 13) and two dummy activities for the source and sink nodes (nodes 1 and 14). The deadline of the project is also considered to be at the end of the 18th period, while the planning horizon is assumed to occur after 25 periods of time.

Project specifications

Figure 2 shows the Activity-n-Node (AoN) presentation of the sample project. This figure shows that the project has 14 nodes. Among these, node numbers 1 and 14 represent dummy nodes for source and sink, respectively. In addition, the yellow nodes (nodes 4, 7 and 10) are OR nodes, while all the other nodes are AND nodes. The parameters of the sample problem are presented in Tables 2 and 3.

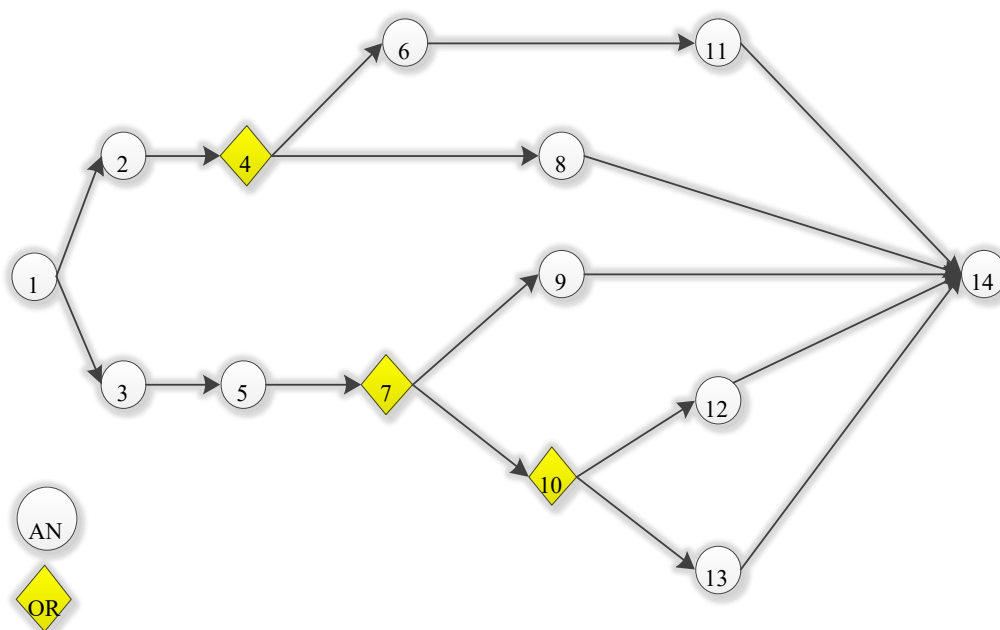


Figure 2. Sample project network.

Table 2. Parameters in the sample problem.

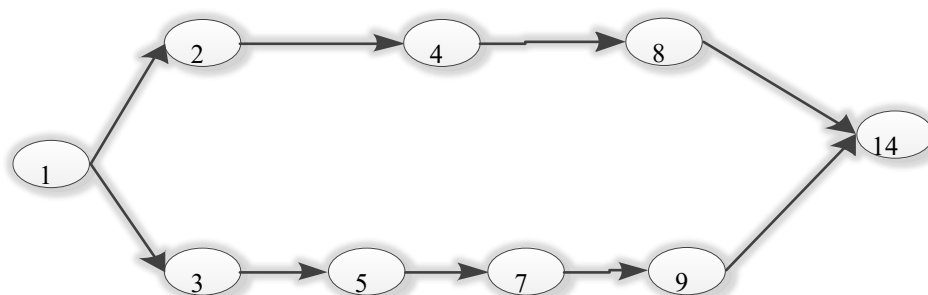
	ACTIVITY											
	2	3	4	5	6	7	8	9	10	11	12	13
c_i	20	20	25	30	30	25	15	20	20	25	30	30
cr_i	21	20	34	39	33	27	17	25	27	32	33	37
T_j^{max}	1	2	2	1	1	1	1	2	1	2	2	2
D_j	2	5	4	3	3	3	3	5	2	4	5	4

Table 3. Resource usage parameters in sample problem.

J	Re_{jr}			U_{ju}		
	1	2	3	1	2	3
2	6	6	6	1	1	1
3	15	15	15	2.5	2.5	2.5
4	12	12	12	2	2	2
5	9	9	9	1.5	1.5	1.5
6	9	9	9	1.5	1.5	1.5
7	9	9	9	1.5	1.5	1.5
8	9	9	9	1.5	1.5	1.5
9	15	15	15	2.5	2.5	2.5
10	6	6	6	1	1	1
11	12	12	12	2	2	2
12	15	15	15	2.5	2.5	2.5
13	12	12	12	2	2	2

Optimal Solution

In the first step, we present the optimal solution obtained by solving the MINLP1 model. In the optimal solution, nodes 1, 2, 3, 4, 5, 7, 8, 9 and 14 are selected for implementation. In addition, nodes 6, 10, 11, 12 and 13 are removed from the network. Figure 3 shows the project network with optimally selected activities. In addition, Figure 4 illustrates optimal scheduling for the sample problem. The value of the objective function in this case is 417.5, and the project is finished in 18 periods of time (just at deadline). Note that no activity is compressed in time because the project could be completed before the deadline and without any tardiness penalty. It is worth noting that there is idleness in the 1st and 7th time periods because the model does not consider any reward for project earliness; thus, there is no motivation for the model to execute the activities without idleness.

**Figure 3.** The project network with optimally selected activities.

The sample problem is also solved by using the MINLP2 formulation. The result of activity selection (variable x_j) is the same as that of the MINLP1 model. Figure 5 shows the Gantt chart of the optimal schedule obtained by solving the sample problem using MINLP2. It can be seen in this figure that the finish time of the project equals 16. However, the objective value is 417.5 again. That is because, in the solution obtained by the MINLP1 formulation (as of MINLP2), there is no delay in project delivery. In addition, no activity time compression is needed to complete the project in 16 periods of time (obtained by solving MINLP2) instead of 18 periods of time (obtained by solving MINLP1); hence, the objective value for solving both the models is the same. In other words, the problem has multiple optimal solutions.

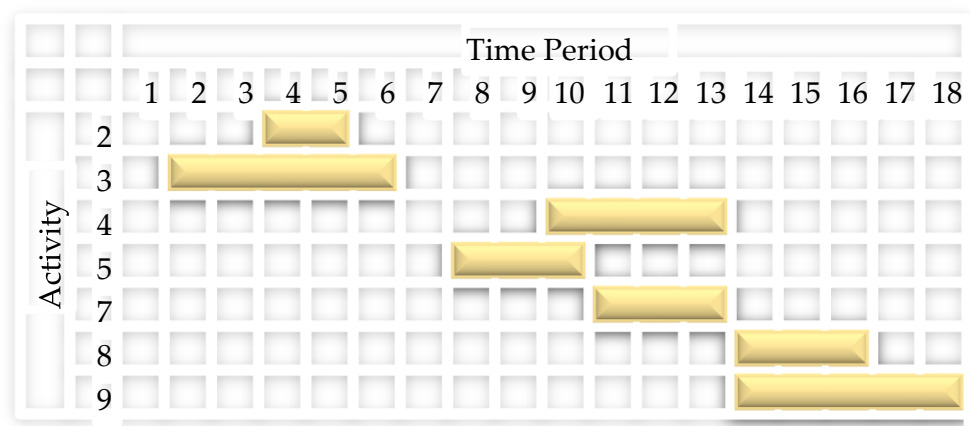


Figure 4. Optimal scheduling for the sample problem (MINLP1).

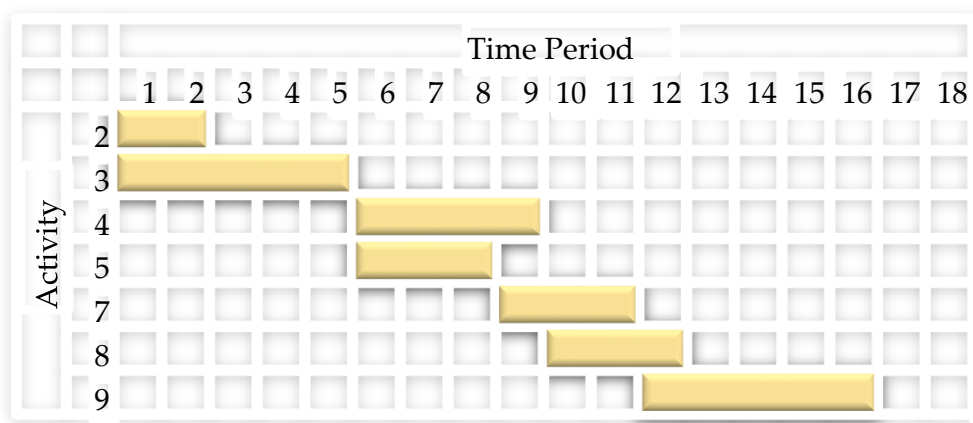


Figure 5. Optimal scheduling for the sample problem (MINLP2).

In addition to MINLP1 and MINLP2, we used a customized GBA to solve the sample problem. In this case, the activity selection decisions (variable x_j), again, result in selection of the same activities as MINLP1 and MINLP2. However, this algorithm yields an objective value of 437.5 (greater than that of MINLP1 and MINLP2). Figure 6 illustrates the schedule obtained by the proposed GBA. This figure shows that the project is completed in 18 periods of time (i.e., with no delay). However, the duration of activity 3 is period-compressed relative to the normal duration (5); hence, the duration of activity 3 in this schedule is 4 periods of time. Activity duration compression results in an increase of 20 units (equal to cr_3) in project cost. Note that this activity compression is not an optimal decision because the project can be completed with no delay penalty with normal activity durations (as scheduled by solving the MINLP1 and MINLP2 models). Although the proposed GBA could not achieve the performance obtained by solving the mixed-integer non-linear programming models (MINLP1 and MINLP2) in solving the sample problem, it may help in solving problems of larger sizes in a more efficient way. Therefore, we examine the computational efficiency of all three solution approaches (MINLP1, MINLP2, and the GBA) in the next section by solving problems with different specifications.

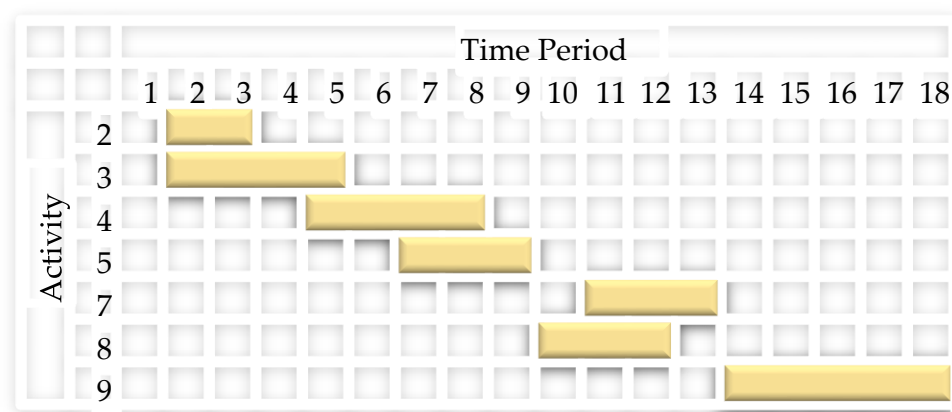


Figure 6. Optimal scheduling for the sample problem (GBA).

4.2. Computational Efficiency

4.2.1. GEKKO vs. Metaheuristic Algorithms

Based on the comprehensive results across 35 PSPLIB instances ranging from 10 to 30 activities (Table 4), the performance comparison between GEKKO, standard GA, PSO, and GBA for solving MINLP1 reveals distinct trade-offs between solution quality, computational efficiency, and scalability. For smaller problems (10–20 activities; instances j102_2 through j207_3), GEKKO demonstrates exceptional performance, consistently finding optimal solutions for all 30 instances, with remarkably low CPU times (averaging 42.8 s, compared to 242.5 s for GA and 241.2 s for GBA), making it consistently faster. For the metaheuristic algorithms, the best solution for each instance is displayed in bold. The GBA proves highly effective, matching GEKKO's optimal solutions for 27 out of 30 problems (90%), with only three instances (j146_6, j141_8, and j181_1) where it fails to achieve optimality, whereas the standard GA achieves optimality for only 17 problems and PSO for none. Notably, for the 12 problems where both the GBA and standard GA attain optimal solutions (highlighted in gray), the GBA demonstrates better CPU efficiency in 9 instances, indicating the repair mechanism not only improves solution quality but also reduces computational overhead.

For larger problems (30 activities; instances j301_1 through j301_5), the scalability limitations of GEKKO become evident, as it successfully solves only j301_2 within the 300 s time limit, while the GBA and standard GA find feasible solutions for all five instances. More importantly, the GBA outperforms the standard GA in 4 out of 5 problems (j301_1, j301_3, j301_4, and j301_5), producing substantially better objectives—for instance, reducing the objective from 641 to 572 for j301_4 (12% improvement) and from 635 to 428 for j301_3 (32.6% improvement)—while the CPU times remain comparable (approximately 300 s for both). This demonstrates that as the problem size increases, the repair mechanism's ability to guide the search toward feasible, high-quality solutions becomes critical, since the standard GA often gets trapped in local optima. The results consistently show that the GBA provides the best overall trade-off: it achieves near-optimal solutions for small problems with competitive computational cost and delivers superior solutions for large problems where exact methods fail, making it the most robust algorithm for practical RCPSP-FNS applications across varying instance sizes.

Table 4. Results of evaluation of computational efficiency.

Problem	Number of Activities	Number of Resources	Objective (GEKKO)	Objective (GA)	Objective (PSO)	Objective (GBA)	CPU Time (GEKKO)	CPU Time (GA)	CPU Time (PSO)	CPU Time (GBA)
j102_2	10	4	169.00	176.00	170.00	169.00	14.34	159.73	197.28	173.81
j102_4	10	4	138.00	161.00	144.00	138.00	14.82	154.62	157.09	163.70
j102_5	10	4	111.00	111.00	113.00	111.00	12.40	165.46	157.61	161.87
j102_6	10	4	119.00	119.00	123.00	119.00	7.69	138.34	127.64	124.55
j102_7	10	4	129.00	129.00	131.00	129.00	24.37	206.92	190.63	215.19
j122_8	12	4	135.00	135.00	153.00	135.00	14.13	187.09	184.47	179.39
j125_9	12	4	153.00	153.00	160.00	153.00	25.06	248.04	223.57	224.46
j128_5	12	4	153.00	192.00	200.00	192.00	20.51	214.48	210.43	208.60
j129_1	12	4	137.00	137.00	160.00	137.00	23.66	207.10	216.49	214.29
j129_2	12	4	129.00	129.00	171.00	129.00	32.91	268.10	262.82	262.65
j141_8	14	4	275.00	301.00	322.00	315.00	29.73	245.57	245.89	249.87
j143_10	14	4	266.00	276.00	381.00	276.00	44.87	270.68	278.08	264.79
j144_10	14	4	236.00	237.00	243.00	236.00	24.73	232.75	248.20	235.23
j146_3	14	4	358.00	358.00	458.00	358.00	28.70	225.44	235.95	226.79
j146_6	14	4	330.00	330.00	891.00	389.00	22.15	199.45	210.06	211.54
j162_3	16	4	385.50	470.00	539.00	470.00	38.65	298.37	298.47	296.78
j162_9	16	4	294.00	416.00	763.00	335.00	37.51	300.01	270.43	276.30
j166_2	16	4	528.00	533.00	801.00	528.00	66.00	300.01	300.04	300.01
j169_1	16	4	330.67	355.00	425.00	355.00	39.16	300.01	292.34	284.17
j169_2	16	4	229.00	419.00	306.00	229.00	46.55	300.01	300.04	300.01
j181_1	18	4	226.50	253.00	431.00	271.00	60.29	300.01	300.04	300.01
j184_6	18	4	213.50	312.00	393.00	260.00	41.45	297.39	300.03	286.63
j188_5	18	4	453.67	688.00	2332.00	579.00	72.78	300.01	300.00	300.01
j189_1	18	4	415.83	530.00	535.00	530.00	67.90	300.01	300.04	300.01
j189_2	18	4	237.00	342.00	333.00	242.00	90.62	300.01	300.04	300.01
j203_2	20	4	300.50	428.00	423.00	332.00	89.61	300.01	300.02	300.01
j203_5	20	4	304.00	430.00	614.00	376.00	40.76	281.86	300.04	277.78
j205_7	20	4	465.00	687.00	588.00	574.00	92.12	300.01	300.04	300.01
j206_3	20	4	482.50	620.00	810.00	612.00	74.28	300.01	300.04	300.01
j207_3	20	4	554.67	605.00	824.00	605.00	182.82	300.01	300.04	300.01
j301_1	30	4	-	428.00	3833.00	428.00	-	300.01	300.05	300.02
j301_2	30	4	416.67	634.00	2178.00	634.00	298.08	300.01	300.05	300.01
j301_3	30	4	-	635.00	1053.00	428.00	-	300.01	300.06	300.01
j301_4	30	4	-	641.00	2099.00	572.00	-	300.01	300.06	300.01
j301_5	30	4	-	492.00	3959.00	519.00	-	300.01	300.06	300.01

The best solution for each instance is displayed in bold. For the 12 problems where both the GBA and standard GA attain optimal solutions, the cells are highlighted in gray.

4.2.2. Comparing MINLP1 and MINLP2

Table 5 compares the results of solving the MINLP1 and MINLP2 models using GEKKO across 35 PSPLIB instances. For the 30 smaller instances (10–20 activities), MINLP2 consistently achieves substantially improved computational efficiency while

maintaining identical objective values in all of 30 instances, with CPU-time improvements ranging from 32% (j102_6) to 88% (j207_3) and averaging approximately 66% faster than MINLP1 (CPU times are visually compared in Figure 7). For example, the solution time of j129_2 improves from 33 s to 7 s (80% reduction), that of j207_3 drops from 183 s to 22 s (88% reduction), and that of j189_2 reduces from 91 s to 15 s (83% reduction).

The most dramatic improvements occur in the 30-activity instances: MINLP1 fails to find any solution within the time limit for four of the five problems (j301_1, j301_3, j301_4, and j301_5) and requires 298 s for j301_2, whereas MINLP2 successfully solves all five instances in just 45–47 s, achieving a complete improvement for the previously unsolvable cases. Statistically compared, the result of a t-test reveals a 0.000 p -value for comparison of the solution times of MINLP1 and MINLP2, which shows that the difference is statistically significant. Overall, the results demonstrate that the MINLP2 formulation dramatically improves both solution speed and reliability, making exact optimization feasible for larger problem instances that were previously intractable.

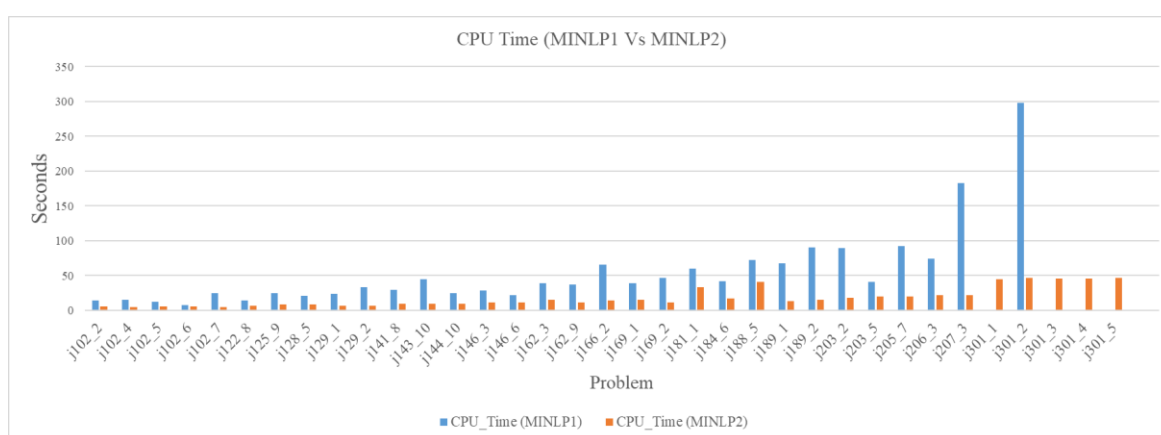


Figure 7. MINLP1 vs. MINLP2 (CPU time).

Table 5. The results of solving the MINLP1 and MINLP2 models using GEKKO.

Problem	CPU Time (MINLP1)	Objective Value (MINLP1)	CPU Time (MINLP2)	Objective Value (MINLP2)	Time Improvement (%)
j102_2	14	169	6	169	61
j102_4	15	138	5	138	66
j102_5	12	111	6	111	53
j102_6	8	119	5	119	32
j102_7	24	129	5	129	79
j122_8	14	135	7	135	52
j125_9	25	153	8	153	68
j128_5	21	153	8	153	59
j129_1	24	137	7	137	72
j129_2	33	129	7	129	80
j141_8	30	275	9	275	70
j143_10	45	266	10	266	79

j144_10	25	236	10	236	61
j146_3	29	358	11	358	60
j146_6	22	330	12	330	47
j162_3	39	386	15	386	60
j162_9	38	294	12	294	69
j166_2	66	528	15	528	78
j169_1	39	331	15	331	62
j169_2	47	229	12	229	75
j181_1	60	226	34	226	44
j184_6	41	213	17	213	59
j188_5	73	454	41	454	44
j189_1	68	416	13	416	81
j189_2	91	237	15	237	83
j203_2	90	300	18	300	80
j203_5	41	304	20	304	52
j205_7	92	465	20	465	78
j206_3	74	483	21	483	71
j207_3	183	555	22	555	88
j301_1	-	-	45	343	-
j301_2	298	417	47	417	84
j301_3	-	-	46	352	-
j301_4	-	-	46	354	-
j301_5	-	-	47	357	-

Based on the comparison of the number of variables between MINLP1 and MINLP2 across all 35 PSPLIB instances (Figure 8), the improved MINLP2 formulation achieves dramatically better CPU performance through substantial reductions in model size and complexity. MINLP2 consistently reduces the number of variables by approximately 15–40% across all instances—for example, from 793 to 673 (15% reduction) for j102_2; from 1681 to 897 (47% reduction) for j129_2; from 2949 to 2113 (28% reduction) for j207_3; and, most notably, from 6913 to 4353 (37% reduction) for problem j301_1. This reduction stems from MINLP2 employing more efficient problem representation techniques, such as reducing the number of time-indexed binary variables (y_{it}) through better scheduling representation. The smaller variable count directly translates to faster solver performance because each iteration of the branch-and-bound or branch-and-cut algorithm requires less memory, fewer floating-point operations, and reduced computational overhead for matrix factorization and linear programming relaxations. This is particularly evident in the 30 activity instances where MINLP1 becomes intractable, with 5761–6977 variables, while MINLP2's streamlined formulation with 4353 variables allows the solver to find solutions within just 45–47 s. The reduced variable count also leads to a smaller search space for the branch-and-bound algorithm, fewer integer combinations to explore, and faster convergence to optimality, ultimately making exact optimization feasible for larger problem instances that were previously unsolvable within practical time limits.

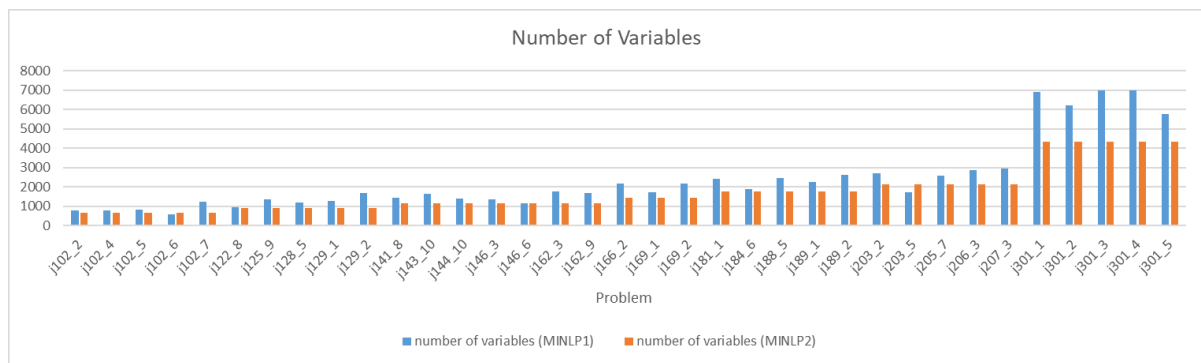


Figure 8. Comparison of the number of decision variables in the MINLP1 vs. MINLP2 formulation.

4.2.3. Sensitivity Analysis on δ

Sensitivity analysis of project cost on δ is examined in this subsection. Based on the sensitivity analysis results across four delta levels (0.05, 0.25, 0.5, and 0.75) in instance 102_2 as an example, increasing the minimum float requirement—which mandates greater schedule flexibility by enforcing larger slack margins for each activity (Figure 9)—drives project costs, jumping sharply from 169 at $\delta = 0.05$ to 228 at $\delta = 0.75$, representing a 35% cost increase when the float requirement becomes too restrictive. The flat cost profile from $\delta = 0.05$ through $\delta = 0.5$ indicates that the schedule naturally provides up to 50% float flexibility without requiring additional resources or compression, meaning the project inherently has sufficient slack to absorb moderate delays at no extra cost. However, the sharp escalation at $\delta = 0.75$ reveals a critical threshold where the system's inherent flexibility is exhausted, forcing the model to adopt more expensive resource allocations, additional activity compression, or extended project durations to maintain the required slack. This demonstrates that while larger delta values provide greater schedule robustness and flexibility to absorb uncertainties and delays, this flexibility comes at a substantial cost penalty only when pushed beyond the natural slack available in the network.

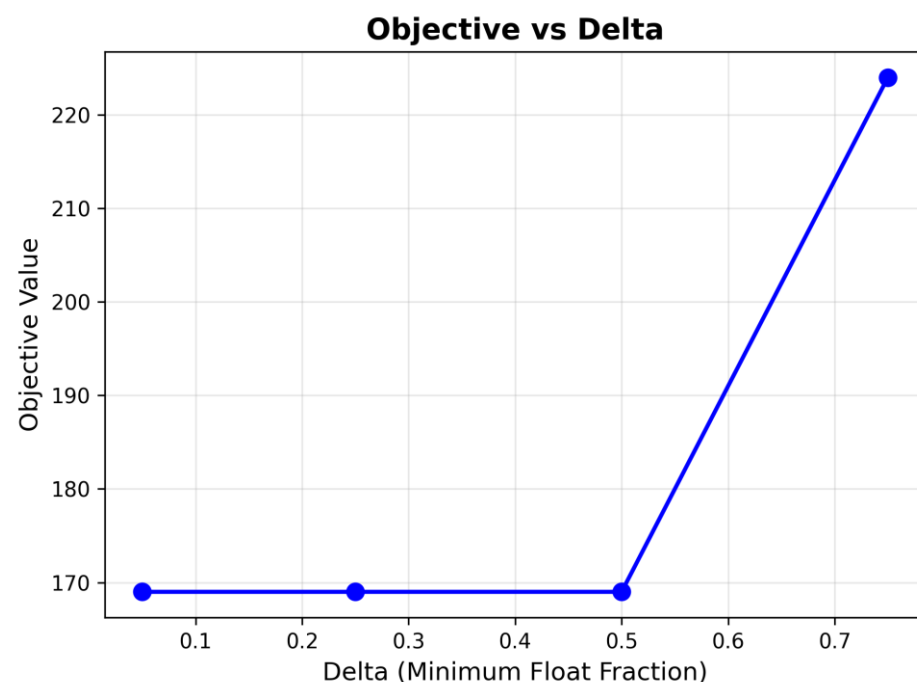


Figure 9. Sensitivity analysis on δ .

4.2.4. Convergence Analysis

Figure 10 illustrates the convergence behavior of the proposed Genetic-Based Algorithm (GBA) across different problems. The figure shows the objective function decreasing from an initial value to a final stabilized value for all the problems. The convergence pattern exhibits a characteristic rapid improvement phase during the first iterations, where the objective function drops sharply, indicating that the algorithm efficiently explores the solution space and quickly identifies promising regions early in the search process. Following this initial phase, the convergence rate gradually decreases, with the objective function declining more slowly, refining the solutions within the identified promising regions. At the last iterations, the GBA achieves near-complete convergence, demonstrating that the algorithm has successfully converged to a high-quality solution with minimal deviation from the optimal solution. This convergence profile confirms the effectiveness of the GBA's hybrid design—combining the global search capability of genetic operators with the local refinement of problem-specific heuristics—in efficiently navigating the complex solution space of the RCPSP-FNS.

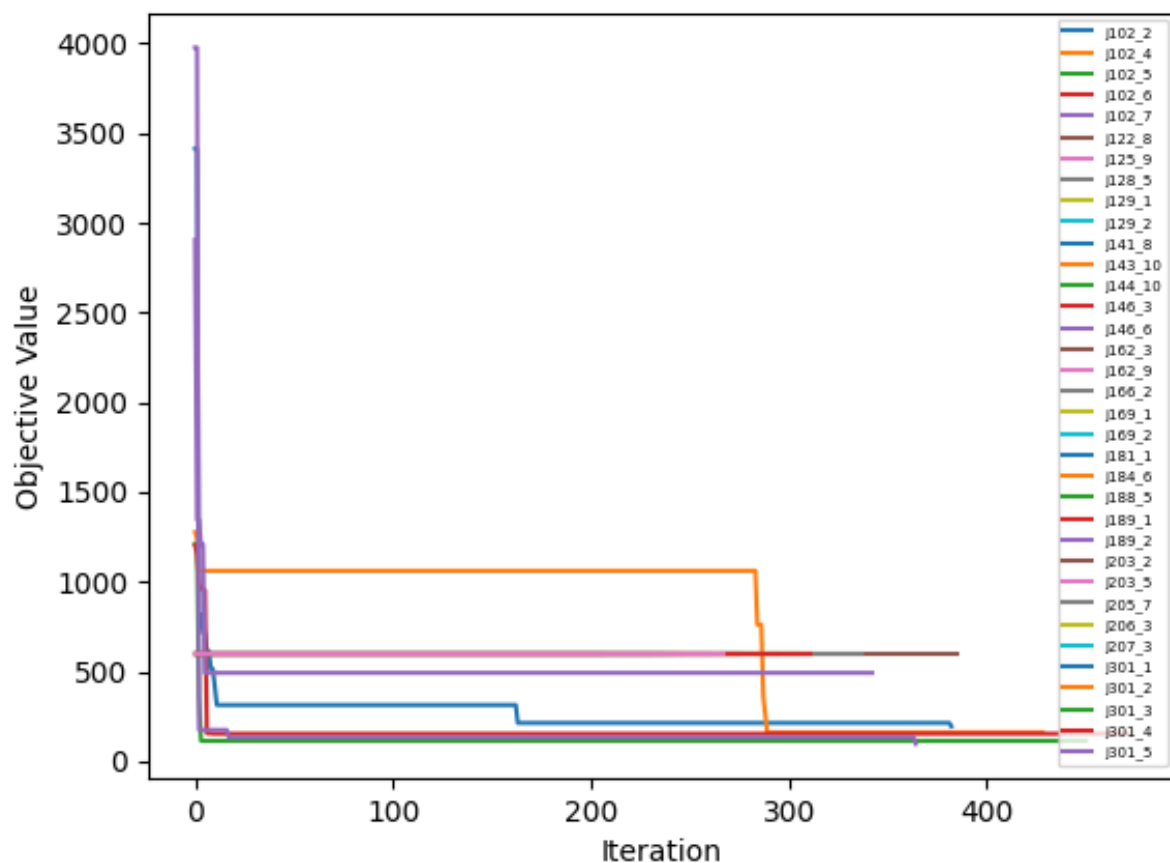


Figure 10. Convergence of the proposed GBA.

5. Conclusions

In this paper, we presented efficient alternative mixed-integer non-linear programming models and a customized hybrid genetic-based algorithm for the resource-constrained project-scheduling problem with a flexible network structure (RCPSP-FNS) with a cost–time trade-off by considering activity duration compression. We used the modeling approach presented by Tao and Dong [4] and presented a mixed-integer non-linear program using binary variables representing activity completion times (called MINLP1) and the GEKKO solver to solve the problem. We also presented a novel alternative mixed-

integer non-linear program using integer variables representing activity completion times (called MINLP2) that has not been presented before in RCPSP-FNS modeling literature. In addition, we proposed a customized hybrid metaheuristic genetic-based algorithm to solve the problems of large size. We evaluated the computational efficiency of the three proposed approaches by solving twenty different problems. The MINLP1 and MINLP2 models were both solved by the GEKKO solver. The results showed that the MINLP2 model strongly outperforms both the MINLP1 model and the proposed genetic algorithm in solving all the examined problems. MINLP2 can be solved by the GEKKO solver to proven optimality in much less time than MINLP1 and the GBA, even for the problem that cannot be solved optimally in a maximum allowable time by MINLP1. The results also showed that the proposed GBA converges very fast towards optimality in first iterations, but it stops at near-optimal solutions. This means that one can use the proposed GBA when a near-optimal solution in a limited time is satisfactory. Future research can focus on solving the problem in uncertain conditions using stochastic modeling. Multi-objective analysis of the problem using modern algorithms like BxR and MO-BxR [38] can also be an attractive research direction. In addition, one of the limitations of this paper is considering only FS precedence relations, which can be improved by modeling generalized precedence relations in future works.

Last but not least, the findings of this study have significant managerial implications for decision-makers and practitioners in RCPSP optimization fields. The analysis and comparison of MINLP1, MINLP2, and the proposed genetic algorithm (GBA) yield valuable insights into their respective performance characteristics.

Author Contributions: Conceptualization, A.P., A.A. and S.S.; Methodology, A.P., and A.A.; Software, S.S.; Validation, A.P., and A.A.; Formal analysis, A.P., and A.A.; Data curation, A.P. and A.A.; Writing—original draft, A.P., A.A. and S.S.; Writing—review & editing, A.P., A.A. and S.S.; Visualization, A.P., A.A. and S.S.; Supervision, A.P. and S.S. All authors made the same contributions. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available at <https://www.om-db.wi.tum.de/psplib/data.php> (accessed on 5 May 2026)

Acknowledgments: The authors acknowledge the use of artificial intelligence as an editorial assistant.

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

Abbreviations

The following abbreviations are used in this manuscript:

GBA	Genetic-Based Algorithm
RCPSP-FNS	Resource-Constrained Project-Scheduling Problem with a Flexible Network Structure

References

1. Mohite, R.; Kanthe, R.; Kale, K.S.; Bhavsar, D.N.; Murthy, D.N.; Murthy, R.D. Integrating Artificial Intelligence into Project Management for Efficient Resource Allocation. *Int. J. Intell. Syst. Appl. Eng.* **2024**, *12*, 420–431.
2. Menon, S. Best Practices and Implementation Challenges in Effective Project Management. *Int. Bus. Res.* **2024**, *17*, 66. <https://doi.org/10.5539/ibr.v17n2p66>
3. Jaafar, K.; Shylo, R. Project Resource Optimization: Considering Labor Productivity Factors. *J. Mod. Proj. Manag.* **2021**, *9*, 34. <https://doi.org/10.19255/jmpm02603>

4. Tao, S.; Dong, Z.S. Multi-mode resource-constrained project scheduling problem with alternative project structures. *Comput. Ind. Eng.* **2018**, *125*, 333–347. <https://doi.org/10.1016/j.cie.2018.08.027>
5. Liu, H.; Qu, S.; Li, R.; Razaa, H. Bi-objective robust project scheduling with resource constraints and flexible activity execution lists. *Comput. Ind. Eng.* **2021**, *156*, 107288. <https://doi.org/10.1016/j.cie.2021.107288>
6. Golab, A.; Gooya, E.; Falou, A.; Cabon, M. A multilayer feed-forward neural network (MLFNN) for the resource-constrained project scheduling problem (RCPSP). *Decis. Sci. Lett.* **2022**, *11*, 407–418. <https://doi.org/10.5267/j.dsl.2022.7.004>
7. Li, H.; Zheng, L.; Zhu, H. Resource leveling in projects with flexible structures. *Ann. Oper. Res.* **2023**, *321*, 311–342. <https://doi.org/10.1007/s10479-022-04797-y>
8. 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. <https://doi.org/10.1016/j.ejor.2021.05.004>
9. Li, H.; Zhang, H. Ant colony optimization-based multi-mode scheduling under renewable and nonrenewable resource constraints. *Autom. Constr.* **2013**, *35*, 431–438. <https://doi.org/10.1016/j.autcon.2013.05.030>
10. Kyriakidis, T.S.; Kopanos, G.M.; Georgiadis, M.C. MILP formulations for single-and multi-mode resource-constrained project scheduling problems. *Comput. Chem. Eng.* **2012**, *36*, 369–385. <https://doi.org/10.1016/j.compchemeng.2011.06.007>
11. Wang, L.; Fang, C. An effective estimation of distribution algorithm for the multimode resource-constrained project scheduling problem. *Comput. Oper. Res.* **2012**, *39*, 449–460. <https://doi.org/10.1016/j.cor.2011.05.008>
12. Zamani, R. A competitive magnet-based genetic algorithm for solving the resource constrained project scheduling problem. *Eur. J. Oper. Res.* **2013**, *229*, 552559. <https://doi.org/10.1016/j.ejor.2013.03.005>
13. Naber, A.; Kolisch, R. MIP models for resource-constrained project scheduling with flexible resource profiles. *Eur. J. Oper. Res.* **2014**, *239*, 335–348. <https://doi.org/10.1016/j.ejor.2014.05.036>
14. Wang, L.; Fang, C. A hybrid estimation of distribution algorithm for solving the resource-constrained project scheduling problem. *Expert Syst. Appl.* **2012**, *39*, 24512460. <https://doi.org/10.1016/j.eswa.2011.08.095>
15. Paraskevopoulos, D.C.; Tarantilis, C.D.; Ioannou, G. Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm. *Expert Syst. Appl.* **2012**, *39*, 3983–3994. <https://doi.org/10.1016/j.eswa.2011.09.062>
16. Xiao, J.; Wu, Z.; Hong, X.X.; Tang, J.C.; Tang, Y. Integration of electromagnetism with multi-objective evolutionary algorithms for RCPSP. *Eur. J. Oper. Res.* **2016**, *251*, 22–35. <https://doi.org/10.1016/j.ejor.2015.10.059>
17. Kreter, S.; Rieck, J.; Zimmermann, J. Models and solution procedures for the resourceconstrained project scheduling problem with general temporal constraints and calendars. *Eur. J. Oper. Res.* **2016**, *251*, 387–403. <https://doi.org/10.1016/j.ejor.2015.11.021>
18. Kellenbrink, C.; Helber, S. Scheduling resource-constrained projects with a flexible project structure. *Eur. J. Oper. Res.* **2015**, *246*, 379–391. <https://doi.org/10.1016/j.ejor.2015.05.003>
19. Cheng, M.Y.; Peng, H.S.; Wu, Y.W.; Chen, T.L. Estimate at completion for construction projects using evolutionary support vector machine inference model. *Autom. Constr.* **2010**, *19*, 619–629. <https://doi.org/10.1016/j.autcon.2010.02.008>
20. Neumann, K. Scheduling with GERT Precedence Constraints. In *Stochastic Project Networks*; Springer: Berlin/Heidelberg, Germany, 1990; pp. 116–175. https://doi.org/10.1007/978-3-642-61515-3_5
21. Tsamardinos, I.; Vidal, T.; Pollack, M.E. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints* **2003**, *8*, 365–388. <https://doi.org/10.1023/A:1025894003623>
22. Beck, J.C.; Fox, M.S. Constraint-directed techniques for scheduling alternative activities. *Artif. Intell.* **2000**, *121*, 211–250. [https://doi.org/10.1016/S0004-3702\(00\)00035-7](https://doi.org/10.1016/S0004-3702(00)00035-7)
23. Ranjbar, M.; Davari, M. An exact method for scheduling of the alternative technologies in R&D projects. *Comput. Oper. Res.* **2013**, *40*, 395–405. <https://doi.org/10.1016/j.cor.2012.07.005>
24. Belhe, U.; Kusiak, A. Resource constrained scheduling of hierarchically structured design activity networks. *IEEE Trans. Eng. Manag.* **1995**, *42*, 150–158. <https://doi.org/10.1109/17.387271>
25. Gillies, D.W.; Liu, J.W.S. Scheduling tasks with AND/OR precedence constraints. *SIAM J. Comput.* **1995**, *24*, 797–810. <https://doi.org/10.1137/S0097539791218664>
26. Čapek, R.; Šůcha, P.; Hanzálek, Z. Production scheduling with alternative process plans. *Eur. J. Oper. Res.* **2012**, *217*, 300–311. <https://doi.org/10.1016/j.ejor.2011.09.018>
27. Birjandi, A.; Mousavi, S.M.; Hajirezaie, M.; Vahdani, B. A new weighted mixed integer nonlinear model and FPNL solution algorithm for RCPSP with multi-route work packages under fuzzy uncertainty. *J. Intell. Fuzzy Syst.* **2019**, *37*, 737–751. <https://doi.org/10.3233/JIFS-181293>

28. Cajzek, R.; Klanšek, U. Cost optimization of project schedules under constrained resources and alternative production processes by mixed-integer nonlinear programming. *Eng. Constr. Archit. Manag.* **2019**, *26*, 2474–2508. <https://doi.org/10.1108/ECAM-01-2019-0013>
29. Van der Beek, T.; Souravlias, D.; van Essen, J.T.; Pruyn, J.; Aardal, K. Hybrid differential evolution algorithm for the resource constrained project scheduling problem with a flexible project structure and consumption and production of resources. *Eur. J. Oper. Res.* **2024**, *313*, 92–111. <https://doi.org/10.1016/j.ejor.2023.07.043>
30. Servranckx, T.; Coelho, J.; Vanhoucke, M. A genetic algorithm for the Resource-Constrained Project Scheduling Problem with Alternative Subgraphs using a boolean satisfiability solver. *Eur. J. Oper. Res.* **2024**, *316*, 815–827. <https://doi.org/10.1016/j.ejor.2024.02.041>
31. Erenguc, S.S.; Ahn, T.; Conway, D.G. The resource constrained project scheduling problem with multiple crashable modes: An exact solution method. *Nav. Res. Logist.* **2001**, *48*, 107–127. [https://doi.org/10.1002/1520-6750\(200103\)48:2<107::aid-nav1>3.3.co;2-0](https://doi.org/10.1002/1520-6750(200103)48:2<107::aid-nav1>3.3.co;2-0)
32. Atli, O. Tabu search and an exact algorithm for the solutions of resource-constrained project scheduling problems. *Int. J. Comput. Intell. Syst.* **2011**, *4*, 255–267. <https://doi.org/10.2991/ijcis.2011.4.2.13>
33. Bruni, M.E.; Pugliese, L.D.P.; Beraldi, P.; Guerriero, F. A computational study of exact approaches for the adjustable robust resource-constrained project scheduling problem. *Comput. Oper. Res.* **2018**, *99*, 178–190. <https://doi.org/10.1016/j.cor.2018.06.016>
34. De Azevedo, G.H.I.; 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.* **2019**, *289*, 809–824. <https://doi.org/10.1016/j.ejor.2019.07.056>
35. Ramos, A.S.; Olivares-Benitez, E.; Miranda-Gonzalez, P.A. Multi-start iterated local search metaheuristic for the multi-mode resource-constrained project scheduling problem. *Expert Syst.* **2022**, *39*, e12830. <https://doi.org/10.1111/exsy.12830>
36. Shuvo, O.; Golder, S.; Islam, M.R. A hybrid metaheuristic method for solving resource constrained project scheduling problem. *Evol. Intell.* **2023**, *16*, 519–537. <https://doi.org/10.1007/s12065-021-00675-x>
37. Zheng, X.L.; Wang, L. A multi-agent optimization algorithm for resource constrained project scheduling problem. *Expert Syst. Appl.* **2015**, *42*, 6039–6049. <https://doi.org/10.1016/j.eswa.2015.04.009>
38. Rao, R.V.; Taler, J.; Taler, D.; Lakshmi, J. A Unified Optimization Approach for Heat Transfer Systems Using the BxR and MO-BxR Algorithms. *Energies* **2025**, *19*, 34. <https://doi.org/10.3390/en19010034>

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.