mathematics

MDPI

Article

A Formulation for the Stochastic Multi-Mode Resource-Constrained Project Scheduling Problem Solved with a Multi-Start Iterated Local Search Metaheuristic

Alfredo S. Ramos [1], Pablo A. Miranda-Gonzalez [2], Samuel Nucamendi-Guillén [1] and Elias Olivares-Benitez [1,*]

[1] Facultad de Ingeniería, Universidad Panamericana, Zapopan 45010, Jalisco, Mexico
[2] Departamento de Ingeniería Industrial, Universidad Católica del Norte, Antofagasta 1270709, Chile
* Correspondence: eolivaresb@up.edu.mx

Abstract: This research introduces a stochastic version of the multi-mode resource-constrained project scheduling problem (MRCPSP) and its mathematical model. In addition, an efficient multi-start iterated local search (MS-ILS) algorithm, capable of solving the deterministic MRCPSP, is adapted to deal with the proposed stochastic version of the problem. For its deterministic version, the MRCPSP is an NP-hard optimization problem that has been widely studied. The problem deals with a trade-off between the amount of resources that each project activity requires and its duration. In the case of the proposed stochastic formulation, the execution times of the activities are uncertain. Benchmark instances of projects with 10, 20, 30, and 50 activities from well-known public libraries were adapted to create test instances. The adapted algorithm proved to be capable and efficient for solving the proposed stochastic problem.

Keywords: stochastic project scheduling; metaheuristic; multiple modes; resource constraints; iterated local search

MSC: 90B36; 90C15; 90C59; 68T20

Citation: Ramos, A.S.; Miranda-Gonzalez, P.A.; Nucamendi-Guillen, S.; Oiveres-Benitez, E. A Formulation for the Stochastic Multi-Mode Resource-Constrained Project Scheduling Problem Solved with a Multi-Start Iterated Local Search Metaheuristic. *Mathematics* **2023**, *11*, 337. https://doi.org/10.3390/math11020337

Academic Editor: Simeon Reich

Received: 13 November 2022
Revised: 22 December 2022
Accepted: 30 December 2022
Published: 9 January 2023

1. Introduction

The multi-mode resource-constrained project scheduling problem (MRCPSP) is considered to be an extension of the resource-constrained project scheduling problem (RCPSP). This problem consists of a set of project activities that need to be scheduled. The activities have finish-to-start precedence relationships, meaning that some activities have to be finished for other activities to start. There is a limited amount of resources, non-renewable and renewable. Each activity has a known duration and requires a certain amount of resources to be completed. In the specific case of the MRCPSP, project activities may be executed according to different modes, with different resource requirements and activity durations determined by each execution mode. According to Blazewicz et al. [1], the deterministic version of the MRCPSP is considered as non-deterministic polynomial-time hard (NP-hard). A feasible solution requires the selection of an execution mode and a start (or finish) time for each activity without exceeding the available resources (of each type) while complying with the relationships of precedence. The objective of the MRCPSP is to find the feasible solution that results in the shortest execution time of the complete project.

This research introduces and solves a stochastic version of this problem, which increases the complexity of the original MRCPSP. In this proposed version, the duration of the activities is uncertain and modeled by means of a stochastic parameter related to a probability distribution, instead of being a deterministic parameter. The possible values of the stochastic parameter are independent for each activity, resulting in many possible scenarios given by different combinations of activity durations. The proposed stochastic

formulation of the problem, along with its corresponding mathematical model, is described in more detail in Section 2.1.

The MRCPSP becomes relevant when managing projects in pursuit of determining a trade-off between the activities' duration and the total resource consumption. In this approach, each activity can accelerate its execution by consuming more resources, or it can be performed with a longer execution time but using fewer resources (with respect to its ordinary execution time).

Regarding the resources, some of them are renewable, meaning that a certain limited amount is available each period (i.e., labor or equipment), while other resources are non-renewable, meaning that a certain limited amount is available for the complete project (the monetary budget is a clear example, since most projects have a determined maximum allotment). To better illustrate our modeling approach, let us consider a couple of activities involved in a construction project. The required setting time for concrete can be accelerated by using a more expensive fast-setting concrete mix, which increases the consumption of a non-renewable resource. The excavation can also be performed in less time by assigning more equipment or workers (renewable resources) to its execution; however, this would result in having fewer equipment or workers available to execute other project activities, which would have to be delayed due to this shortage of resources, probably resulting in a longer total completion time of the project. This trade-off shows the complexity of this problem, which is also present in several other business or industrial sectors such as cargo transportation, software development, and medical diagnosis [2]. Thus, designing and implementing efficient and reliable methods for solving the MRCPSP becomes relevant. Since there is usually some degree of uncertainty in those industrial and business cases, it is useful to have a capable method to solve stochastic versions of the problem, and more specifically, when the duration of the activities is uncertain.

Small instances of the deterministic version of the MRCPSP have been successfully solved via exact mixed-integer linear programming (MILP) methods [3], such as branch and cut [4] or branch and bound [5,6]. However, problem instances involving three execution modes per activity and more than twenty project activities cannot be solved using exact methods in a reasonable computational time [7], arising the convenience of employing heuristic and metaheuristic methods to obtain high-quality solutions for large instances of the MRCPSP or to provide solutions for its stochastic version.

This study introduces a stochastic version of the MRCPSP with uncertain activity duration, and proposes an efficient solution approach by adapting a multi-start iterated local search (MS-ILS) metaheuristic originally developed by Ramos et al. [8] for solving the deterministic version of the problem. The iterated local search strategy works in the following way: first, an initial solution is generated and a local search is applied to it to obtain a local optimum; then for a given number of iterations, a perturbation move is applied to the local optimum, a new local search is performed, and a new local optimum is found [9]. After all of the iterations have been completed, the algorithm restarts, generating a new initial solution that is unrelated to the previous ones. After a certain number of restarts, the algorithm stops, providing the best solution that has been found in all the iterations of all the restarts.

1.1. Literature Review

The resource-constrained project scheduling problem (RCPSP) is a well-known optimization problem with the objective of minimizing the total duration of a project by scheduling its activities, complying with resource and precedence constraints [10–14]. With respect to the multi-mode resource-constrained project scheduling problem (MRCPSP), it was introduced by Elmaghraby [15] as a generalization of the RCPSP. Talbot [16] presented its first mathematical model, and as aforementioned, it was classified as NP-hard by Blazewicz et al. [1]. According to Kolisch and Drexl [17], when the problem involves several types of renewable resources, just finding feasible solutions becomes by itself an NP-complete optimization problem.

To address the deterministic version of the MRCPSP, different authors over the last few decades have developed diverse solution methods involving heuristic or metaheuristic algorithms, and have provided, in some cases, optimal solutions. The proposed approaches include population-based schemes, such as: genetic algorithms [18–24], particle swarm optimization [25,26], scatter search [27,28], differential evolution [29], evolutionary algorithms [30], estimation of distribution algorithms [31], ant colony optimization [32,33], multi-agent learning [34]; or trajectory based algorithms, such as: simulated annealing [35–37], path-relinking [38], and multi-start iterated local search [8].

Regarding stochastic versions of the MRCPSP, to the best of the authors' knowledge, only a few published studies have been found in the literature. Chen and Zhang [39] proposed an ant colony system with Monte Carlo simulation to solve a stochastic MRCPSP with discounted cash flows, with the objective of maximizing the expected net present value of the cash flows of the project, given uncertain activity durations and costs. They considered uniform distributions for the duration to follow and the cost. Chakrabortty and Ryan [40] proposed a robust optimization scheme for the MRCPSP in order to minimize the project makespan. The robust optimization framework is based on a modified version of the variable neighborhood search heuristic, and demonstrated its effectiveness by successfully solving data instances up to 30 activities and a real-life project involving 25 activities. Balouka and Cohen [41] developed a robust optimization approach using Benders decomposition with specialized cuts to solve the MRCPSP with uncertain activity durations that vary within polyhedral uncertainty sets with the objective of minimizing the worst-case project duration. Xie et al. [42] defined an MRCPSP with uncertain activity cost which follows a normal distribution with the objective of minimizing the risk of exceeding a given budget for the entire project. They developed a hybrid construction heuristic and genetic algorithm method to solve it. Each author proposed a different version of a stochastic MRCPSP, with varying stochastic parameters and probability distributions associated with them, as well as different approaches to solving it. Regarding the MRCPSP involving multiple objectives, Azimi and Sholekar [43] proposed a simulation-based approach to solve a biobjective problem that seeks to minimize both project makespan and the present value costs. The algorithm operated in four steps: (1) Decision variable relaxation and the transmutation of the multi-objective problem to a linear single objective model, (2) Solving the new linear model, (3) Simulating the model using the results of the linear model, and (4) Solution fixing. The proposed scheme also solved instances up to 30 activities in reasonable computational times. Another biobjective approach has been recently presented by Yuan et al. [44], who studied a prefabricated building construction project that consider activities with uncertain duration times with the purpose of seeking the best trade-off between the project makespan and its respective operational cost. As a solution method, the authors present a hybrid cooperative co-evolution algorithm that incorporated a self-adaptive mechanism and a self-adaptive selection process (execution mode for the activities). Their experimental results indicate that the proposed algorithm outperformed some existing methods from the literature, previously developed for similar problems, for both the quality of the solution and the elapsed computational time when solving benchmark datasets involving up to 90 activities for the single-mode version of the problem, and up to five different execution modes for datasets of 18 activities of the multi-mode version of the problem.

In our proposed stochastic MRCPSP, the stochastic parameter is the activity duration which follows a discretized triangular distribution [45]. Discrete triangular distributions have been proposed for stochastic parameters of other problems, such as the RCPSP (single mode) [46,47], the capacity-constrained supplier selection model with lost sales under stochastic demand behavior [48], the single machine maximum lateness stochastic scheduling problem [49], the stochastic Vehicle Routing Problem with Restocking [50], and the vehicle routing problem with hard time windows, and stochastic travel and service time [51]. The stochastic version of the MRCPSP with uncertain activity duration, which follows an approximately discrete triangular distribution, is proposed for the first time in

this research. We aim to contribute to filling the gap in the literature since only four research papers were found regarding stochastic versions of the MRCPSP, each one proposing a different one.

In recent years, multi-start iterated local search (MS-ILS) algorithms have been proposed as a solution method for different optimization problems, including the periodic vehicle routing problem [52], the two-echelon routing problem [53], the generalized quadratic multiple knapsack problem [54], the mixed fleet vehicle routing problem [55], the covering salesman problem [56], the uncapacitated single allocation hub location problem [57], and the capacitated vehicle routing problem [58]. More specifically, ref. [8] proposed for the first time an MS-ILS algorithm as a method for solving the deterministic version of the MRCPSP, and they obtained good results. This research aims to adapt that algorithm to solve the proposed stochastic MRCPSP with uncertain activity duration.

To evaluate the effectiveness of the adapted MS-ILS metaheuristic to solve the proposed stochastic MRCPSP, subsets of publicly available benchmark instances for the deterministic version of the problem were adapted to create stochastic instances and to carry out computational experiments. The deterministic instances were taken from the MMLIB [59] and the PSPLIB [60] libraries.

This research provides the following main contributions:

- A previously unstudied stochastic version of the MRCPSP with uncertain activity duration was proposed and formulated, along with its mathematical model.
- A recently developed MS-ILS metaheuristic to solve the deterministic version of the MRCPSP was successfully adapted to solve the proposed stochastic problem.
- The adapted MS-ILS method proved to be capable and efficient for solving the aforementioned problem.

The remainder of this research is presented as follows. Section 2 describes the mathematical formulation for the proposed stochastic MRCPSP, as well as the MS-ILS metaheuristic algorithm developed by Ramos et al. [8] for solving the deterministic MRCPSP, and adapted in this research to solve a stochastic version of the problem. It also explains the computational experiments developed to assess the performance of the algorithm. In Section 3, the results obtained from the computational experiments are presented and discussed. Section 4 presents the conclusions of this research.

2. Materials and Methods

2.1. Problem Description and Mathematical Model

This section describes the proposed stochastic multi-mode resource-constrained project scheduling problem, shows its complexity, and introduces the mathematical formulation to represent it.

The multi-mode resource-constrained project scheduling problem (MRCPSP) can be formally defined as the set ACT of project activities to be scheduled. These activities are labeled as $i = 0, \ldots, I+1$, where activity 0 is a dummy "start" activity, activities 1 to I are the real project activities, and activity $I+1$ is a dummy "finish" activity. The set PRE contains the relationships of precedence between pairs of project activities, defined as the arcs $(i, j) \in PRE$, where i is an immediate predecessor of j. All of the immediate predecessors of any activity must be finished before such an activity may start. Each project activity has an activity number that is greater than those of all its predecessors, making the set ACT topologically ordered. Figure 1 shows an example project involving four real activities (and the corresponding dummy activities), represented as a network diagram where the nodes represent project activities and the arrows represent relationships of precedence, which are given by $PRE = \{(0,1), (0,2), (1,4), (2,3), (3,4), (4,5)\}$.

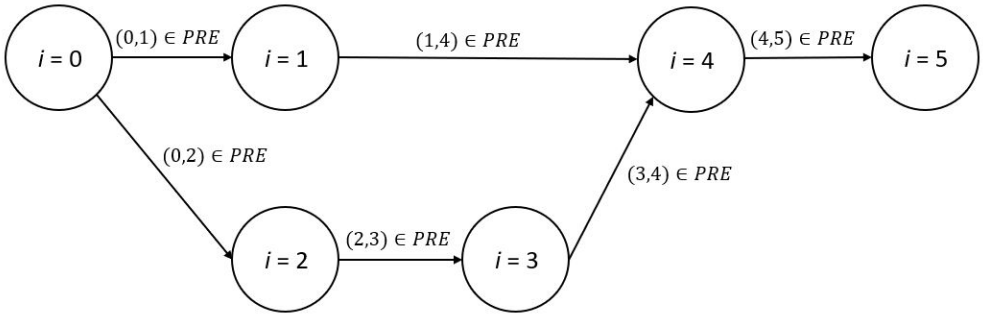

Figure 1. Network diagram of a project.

The execution of each activity requires a certain amount of resources, either non-renewable, renewable, or both. Non-renewable resources can be of R_{nr} types, belonging to the set NON, and labeled as $nr = 1, \ldots, R_{nr}$, while renewable resources can be of R_r types, belonging to the set REN, and labeled as $r = 1, \ldots, R_r$. There is a limited availability of each type of non-renewable resource for the whole project given by the parameter K_{nr}, and a limited availability of each type of renewable resource for each time period (being refilled at the beginning of the next period) given by the parameter K_r. There is a set MOD with M different execution modes for the activities, labeled as $m = 1, \ldots, M$, which determine the resource consumption and the duration of each activity. The amount that activity i executed in mode m requires of each type of non-renewable resource nr is given by the parameter $U_{im(nr)}$, and the amount that activity i executed in mode m requires of each type of renewable resource r per time period t is given by the parameter U_{imr}. The duration of activity i executed in mode m is given by the parameter d_{im}. Both dummy activities require no amount of resources of any type, and the duration of both is 0 time units. The parameter EF_i provides a lower bound for the finish time of activity i (earliest finish), and the parameter LF_i provides an upper bound for its finish time (latest finish) [21].

The main difference between the proposed stochastic MRCPSP and its deterministic version is that the duration of the project activities implies uncertainty. Thus, the parameter d_{im} (duration of activity i in mode m), instead of being deterministic, is redefined as a stochastic parameter $d_{im(k)}$ whose value is given by a discrete random variable with K possible realizations, each one with an associated probability $P_{(k)}$ of occurrence. The value of K is the same for all activities, and the values of $k = 1, \ldots, K$ are considered to be independent for each activity i and each mode m, resulting in independent values of $d_{im(k)}$, meaning that for an instance with I activities and M modes, there are $S = K^{IM}$ possible scenarios, one for each different combination of values of $d_{im(k)}$.

The objective of the problem is to select the execution mode m of all activities i to minimize the expected value of the total project duration over all evaluated scenarios S. This can be obtained by scheduling the activities for each evaluated scenario $s = 1, \ldots, S$, considering their durations and precedence relationships. This solution represents a baseline schedule. The decision variable x_{imts} takes a value of 1 if activity i is executed in mode m, and finishes at time period t in the scenario s, and a value of 0 otherwise; the parameter d_{ims}, duration of activity i in mode m in scenario s replaces the parameter d_{im} from the deterministic version of the problem; and an auxiliary binary variable y_{im} with a value of 1 if the activity i is executed in mode m, and a value of 0 otherwise is also added to the model.

The trade-off between the duration and the resource requirement of the activities makes this problem a complex one. The decision to execute an activity in its mode with the shortest duration may result in a longer duration of the complete project, since that activity would require more resources, which would leave fewer resources available for other activities, resulting in the need for delaying them or executing them in a mode with a longer duration.

The sets, parameters, and variables required for the mathematical formulation of the proposed stochastic MRCPSP with uncertain activity duration are the following:

Sets:

- ACT: project activities $i = 1, \ldots, I$.
- MOD: execution modes $m = 1, \ldots, M$.
- TIM: time periods $t = 1, \ldots, T$.
- PRE: relationships of precedence (i, j). Set of arcs (i, j), meaning activity i is an immediate predecessor of activity j.
- REN: types of renewable resources: $r = 1, \ldots, R_r$.
- NON: types of non-renewable resources: $nr = 1, \ldots, R_{nr}$.
- SCE: probabilistic scenarios.

Parameters:

- EF_i: earliest possible finish time of activity i.
- LF_i: latest possible finish time of activity i.
- K_r: renewable resources availability.
- K_{nr}: non-renewable resources availability.
- d_{ims}: duration of activity i executed in mode m in scenario s.
- U_{imr}: amount of renewable resource r required by activity i executed in mode m for each period of time t (this amount is the same for each period of time).
- $U_{im(nr)}$: amount of nonrenewable resource nr required by activity i executed in mode m.
- P_s: probability of scenario s.

Variables:

- X_{imts}: binary decision variable that takes a value of 1 if activity i is executed in mode m and finishes at time t in scenario s, and a value of 0 otherwise.
- Y_{im}: auxiliary binary variable that takes a value of 1 if activity i is executed in mode m, and a value of 0 otherwise.

The mathematical formulation for the proposed stochastic MRCPSP with uncertain activity duration (based on the deterministic formulation in [8]) is the following:

Minimize:

$$\sum_{s=1}^{S} P_s \sum_{t=EF_{I+1}}^{LF_{I+1}} t x_{I+1,1,t,s,} \tag{1}$$

subject to:

$$\sum_{m=1}^{M_i} \sum_{t=EF_i}^{LF_i} x_{imts} = 1 \qquad \forall i \in ACT, \forall s \in SCE \tag{2}$$

$$\sum_{m=1}^{M_i} y_{im} = 1 \qquad \forall i \in ACT \tag{3}$$

$$\sum_{t=EF_i}^{LF_i} x_{imts} = y_{im} \qquad \forall i \in ACT, \forall s \in SCE, \forall m \in MOD \tag{4}$$

$$\sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} (t - d_{jms}) x_{jmts} \geq \sum_{m=1}^{M_i} \sum_{t=EF_i}^{LF_i} t x_{imts} \qquad \forall (i, j) \in PRE, \forall s \in SCE \tag{5}$$

$$\sum_{i=1}^{I} \sum_{m=1}^{Mi} U_{imr} \sum_{q=t}^{t+d_{ims}-1} x_{imqs} \leq K_r \qquad \forall r \in REN, \forall t \in TIM, \forall s \in SCE \tag{6}$$

$$\sum_{i=1}^{I} \sum_{m=1}^{Mi} U_{im(nr)} \sum_{t=EF_i}^{LF_i} x_{imts} \leq K_{nr} \qquad \forall nr \in NON, \forall s \in SCE \tag{7}$$

$$x_{imts} \in \{0, 1\}, y_{im} \in \{0, 1\}, m = 1, \ldots, M, i = 1, \ldots, I, t = EF_0, \ldots, LF_{I+1}, s = 1, \ldots, S. \tag{8}$$

The objective function (1) aims to minimize the expected value for all evaluated scenarios of the completion time of a "finish" dummy activity I+1, and therefore, the expected total project duration. Constraints set (2) guarantee that, for each scenario, all the activities are scheduled considering an execution mode and are assigned a finish time between their EF and LF times. Constraints (3) and (4) ensure that each activity is executed in only one mode, and that the mode chosen for each activity is the same for each scenario. Constraints (5) establish that, for each activity j, it must start after its immediate predecessors' i is finished. Constraints (6) make sure that the sum of renewable resources r being consumed at each time t does not exceed availability K_r. Constraints (7) ensure that, for each non-renewable resource nr, its total consumption must not exceed their respective availability K_{nr}.

The mathematical complexity of this proposed stochastic MRCPSP with uncertain activity duration is greater than that of the deterministic version, since the inclusion of the scenarios significantly increases the number of possible schedules. Specifically, in the deterministic version, there are M^I possible different solutions, one for each combination of modes m assigned to activities i. In the case of the proposed stochastic formulation of the problem, the number of possible solutions is multiplied times the number of scenarios to evaluate, since there are K^{IM} possible scenarios. If all of them were evaluated, there would be $K^{IM}(M^I)$ possible schedules.

2.2. Multi-Start Iterated Local Search (MS-ILS) Methodology

This section details the components of the multi-start iterated local search method developed by [8] for the deterministic version of the multi-mode resource-constrained project scheduling problem (MRCPSP) and adapted in this research for the proposed stochastic version of the problem with uncertain activity duration.

2.2.1. General Methodology

The general procedure for the proposed multi-start iterated local search (MS-ILS) metaheuristic is described next. A feasible initial solution is generated by randomly selecting the modes of each activity and by scheduling the activities according to those modes for a certain number of scenarios, followed by a local search applied to the initial solution. Later, a perturbation move is performed on the previously obtained local optimum for a certain number of iterations. The perturbed solution is improved using local search again. These steps are repeated until a stopping criterion is met. All of this process is restarted a number of times to produce different solutions, and the best solution of these restarts is the output of the metaheuristic. A flowchart of the description explained above is presented in Figure 2.

The proposed codification structure for the solutions of the stochastic MRCPSP is as follows:

1. X_{imsbf} variables are stored in a matrix with I+2 rows and 4 columns. The rows correspond to each i activities $(0 \leq i \leq I+1)$, where I denotes the number of activities in the project, a dummy "start" activity i = 0 is at the first row, and the last row is for the dummy "finish" activity, i.e., activity I+1. The first column indicates the mode m $(1 \leq m \leq M)$ of each activity, where M denotes the number of modes to perform an activity. The second column indicates the scenario s $(1 \leq s \leq S)$, where S denotes the number of different scenarios to be evaluated. The third column stores the start time b of each activity. The last column stores the finish time of each activity $f = b + d_{ims}$, with d_{ims} being the duration of activity i executed in mode m in scenario s.
2. The objective function is stored in another variable, whose value will be equal to the expected value for all evaluated scenarios of the finish time f of the dummy activity $I+1$.
3. Feasibility is stored in a binary variable that takes the value of 1 when the solution is feasible, and the value of 0 otherwise.

The metaheuristic works with feasible solutions only. The non-renewable resource consumption is added after all activities have their modes settled. If this sum is greater than the available non-renewable resources, the activity mode must be adjusted. Once this constraint is met, the scheduling of the activities is calculated, activity-by-activity. Each activity starts as early as possible, considering the precedence constraints and the limits on the renewable resources consumption.

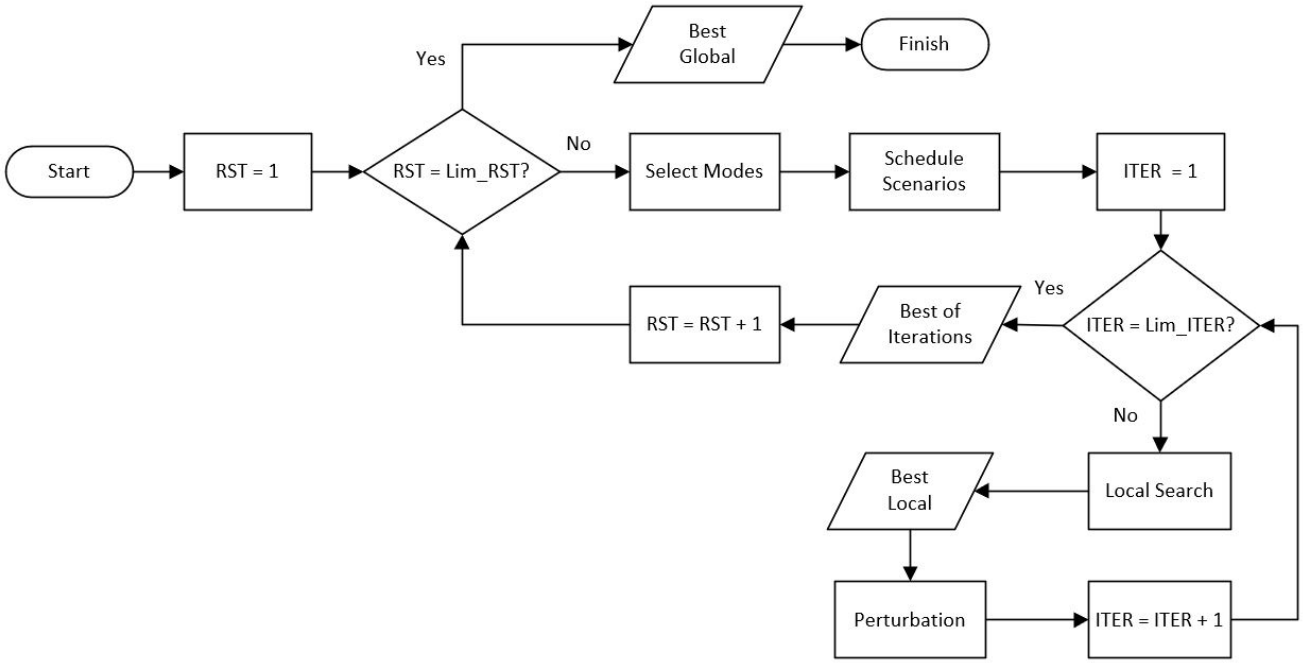

Figure 2. Flowchart of the multi-start iterated local search. ITER = Iterations, RST = Restarts.

2.2.2. Initial Solution Generation

The search space for an instance of this problem is determined by M, the number of execution modes, and I, the number of activities. That is, there are M^I possible solutions. Nevertheless, since the resource constraints induce some solutions to be infeasible, in highly constrained instances, a reduced portion of the solutions are feasible. Regarding the x_{imts} variables, the computation of the number of variables is the multiplication of set sizes M, I, S, and the time length. For example, an instance with 50 activities, 3 modes per activity, 20 scenarios, and a time length of 200 units, will produce a model with 600,000 binary variables. The huge size of the search space and the number of variables that this problem has, require a challenging computational time and resources to solve it. Therefore, the convenience of using heuristic or metaheuristic methods to solve them is justified by this complexity. According to [17], referring to the deterministic case of the problem, when two or more non-renewable resources are available, the task of finding feasible solutions represents an NP-complete problem.

An initial solution within the search space is obtained with the procedure shown in Algorithm 1. It starts by selecting a mode m for each activity i until a feasible solution is found. An adaptive heuristic procedure is used to find a feasible solution. The heuristic begins selecting, for each activity, a random execution mode. If, after a number of iterations only infeasible initial solutions are found, the algorithm randomly chooses a value of 0 or 1 for each activity (RAND {0,1}). The activities with a value of 0 receive a random mode. For those activities with a value of 1, the mode with the lowest sum of nonrenewable resource requirements is selected. If, only infeasible solutions are obtained after a number of additional iterations, the algorithm randomly selects a value of 0, 1, or 2 for each activity (RAND {0,1,2}). The modes of the activities with a value of 0 are selected randomly, and the activities with a value of 1 or 2 receive the mode with the lowest sum of nonrenewable

resource requirements. If after a certain number of additional iterations, only infeasible solutions are still found, the algorithm stops. If a feasible solution is found, the output of the algorithm is the selection of modes for each activity i, represents as a vector with the mode m selected. (See Algorithm 1, algorithm Select_Modes). This procedure was demonstrated in practice to be effective in finding a feasible combination of modes, because none of the tests exceeded the limit of iterations without finding a feasible solution.

Algorithm 1: Select_Modes. Algorithm for the selection of modes.

Input: $U_{im(nr)}, K_{nr}$
Initialize Solution_counter = 1;
repeat
  if *Solution_counter* $\leq$ *CounterLimit_1* **then**
    for i = *1 to* i = I **do**
      Select randomly a mode m for activity i; $i = i + 1$;
  if *CounterLimit_1* $<$ *Solution_counter* $\leq$ *CounterLimit_2* **then**
    for i = *1 to* i = I **do**
      $a \leftarrow$ RAND{0,1};
      if a = *0* **then**
        Select randomly a mode m for activity i;
      else
        Select the mode m with the lowest non-renewable resource requirements for activity i;
      $i = i + 1$;
  if *CounterLimit_2* $<$ *Solution_counter* $\leq$ *CounterLimit_3* **then**
    for i = *1 to* i = I **do**
      $a \leftarrow$ RAND{0,1,2};
      if a = *0* **then**
        Select randomly a mode m for activity i;
      else
        Select the mode m with the lowest non-renewable resource requirements for activity i;
      $i = i + 1$;
  if *Solution_counter* $>$ *CounterLimit_3* **then**
    Output: "No feasible solution found"
    End Algorithm;
  Solution_counter = Solution_counter + 1;
until $\sum U_{im(nr)} \leq K_{nr} \quad \forall nr \in NON$; `/* Non-renewable resource constraints satisfied */`
Output: Mode m selected for each activity i

When each activity has a selected mode (using the Select_Modes algorithm), the activities are sequentially scheduled for each scenario s to be evaluated, employing a serial schedule generation scheme (SGS) [61]. Each activity is scheduled to start at the earliest time period t possible, satisfying the precedence constraints and not exceeding the limit on renewable resources. This generates an active schedule where all activities in each scenario are scheduled as early as possible, as required by [62]. An initial feasible solution with the objective function value calculated as the expected value of the project finish time considering all evaluated scenarios and their probabilities, is obtained. (See Algorithm 2, algorithm Schedule_Scenarios).

Algorithm 2: Schedule_Scenarios. Algorithm for solution scheduling for the scenarios.

Input: Selection of modes from algorithm Select_Modes
for s = *1 to S ;* `/* For each scenario */`
do
  Schedule dummy activity 0 to start at t = 0 and finish at t = 0;
  for i = *1 to* i = *I+1* **do**
    Schedule activity i in scenario s to start at time b immediately after all its predecessors' finish;
    b_{is} = start time of activity i in scenario s;
    $f_{is} = b_{is} + d_{ims}$; `/* Finish time of activity` i `in scenario` s `*/`
    if $\sum_{t=b}^{f} U_{imr} > K_r$ **then**
      repeat
        Reschedule activity i to start at $b + 1$;
        $b_{is} = b_{is} + 1, f_{is} = b_{is} + d_{ims}$;
      until $\sum_{t=b}^{f} U_{imr} \leq K_r \quad \forall r \in REN$; `/* Renewable resource constraints satisfied */`
    $i = i + 1$;
  $OF_s = f_{I+1,s}$; ; `/* Objective function of the scenario = finish time of last activity */`
$OF = \sum_{s=1}^{S} OFsPs$; `/* Objective function = expected value of all` OF_s `*/`
Output: Solution *Sol*

To exemplify the procedure shown in Algorithm 2, the project schematized in Figure 1, along with the information in Table 1, is taken as an example, considering only one mode and one scenario, and taking into account the following way to graphically represent the project (shown in Figure 3): each gray block depicts an activity with its number; the width of the block symbolizes the activity duration for a certain execution mode, and the height symbolizes the consumption of the renewable resource by the activity on that execution mode. The activities are scheduled as early as their immediate predecessors are finished when sufficient renewable resources are available. In this case, two units of one renewable resource are available per time period. The steps required to schedule the activities of this example project are presented in Figure 3, and describe the following. First step: activity 1 can be scheduled to start at time $t = 0$ since it has no predecessors, with the exception of the dummy activity, and it finishes at time $t = 4$. It consumes one unit of renewable resource per time period. Second step: activity 2 is also scheduled to start at time $t = 0$ because it has no predecessors, with the exception of the dummy activity. This activity consumes 1 unit of renewable resource per time period and finishes at time $t = 3$. During the first three time periods, both activities 1 and 2 are active, consuming each one 1 unit of the renewable resource, 2 units in total, which is the available limit. Third step: activity 3 has only one predecessor which is activity 2, which means that it could start as soon as activity 2 finishes at time $t = 3$. However, since activity 3 requires 2 units of the renewable resource, and activity 1 is already consuming 1 unit during period 4, activity 3 can start only at time $t = 4$, when the 2 units of the renewable resource become available. Fourth step: The sooner activity 4 can start is when both its predecessors (activities 1 and 3) are finished, which is at time $t = 6$. The finish time of activity 4 is a time $t = 7$, which is also the finish time of the complete project, with a total duration of seven time units. The resulting schedule is shown in Table 2.

Table 1. Parameters of a sample project.

Activity	Predecessors	Mode	Duration	Consumption of Renewable Resource
1	0	1	4	1
2	0	1	3	1
3	2	1	2	2
4	1, 3	1	1	1

Table 2. Schedule of a sample project.

Activity	Mode	Duration	Start Time	Finish Time
1	1	4	0	4
2	1	3	0	3
3	1	2	4	6
4	1	1	6	7

The Select_Modes algorithm for selecting the modes, and the Schedule_Scenarios algorithm for scheduling the activities are repeated at each restart of the metaheuristic to create a starting solution.

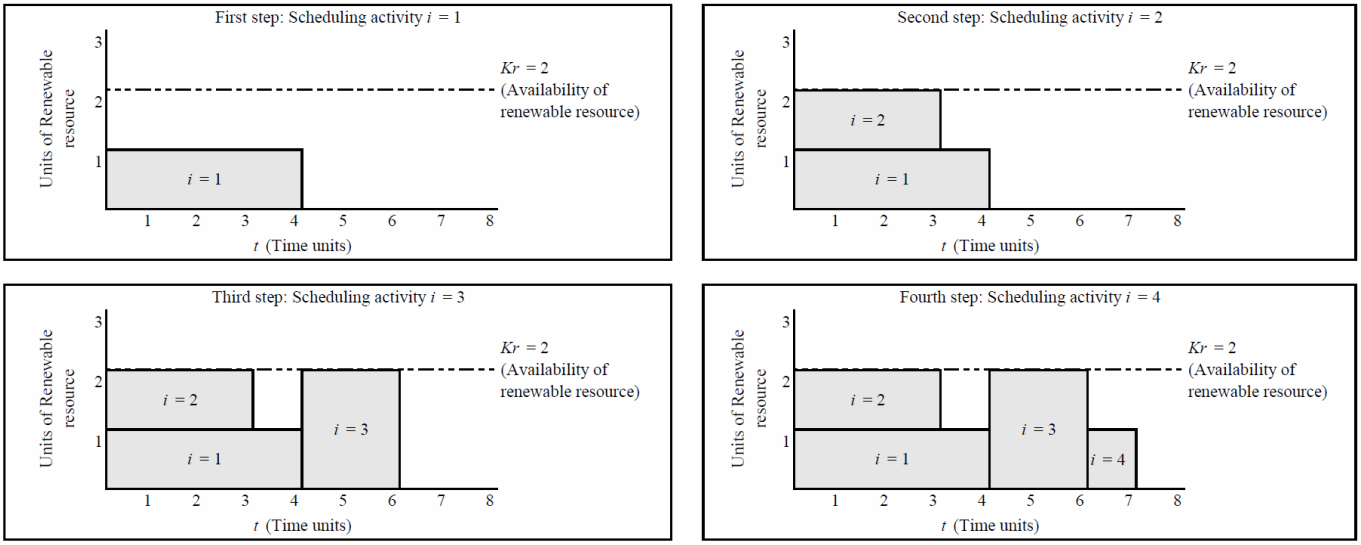

Figure 3. Step by step scheduling of a solution for a sample project following a serial schedule generation scheme (SGS) [61].

2.2.3. Neighborhood Structure and Local Search

The local search of the ILS explores a neighborhood structure that is defined by all the solutions with one and only one activity executed in a different mode with respect to the original solution. The neighborhood structure exhibited in Table 3 shows an example. The idea is to produce a short-range neighborhood that intensifies the local search. All neighbors are explored using a descent local search, changing randomly the mode of one activity at a time. The change of mode is shown in bold/italics for every neighbor solution. For each new solution, the objective function and feasibility are evaluated, and the feasible solution with the shortest expected value of the project duration is stored as the new solution and the local optimum (see Algorithm 3, algorithm Local_Search).

Table 3. Neighborhood structure example.

Solution S		Neighbor S_1		Neighbor S_2		Neighbor S_3		Neighbor S_4	
Activity	**Mode**	**Activity**	**Mode**	**Activity**	**Mode**	**Activity**	**Mode**	**Activity**	**Mode**
1	2	1	***1***	1	2	1	2	1	2
2	1	2	1	2	***2***	2	1	2	1
3	1	3	1	3	1	3	***2***	3	1
4	2	4	2	4	2	4	2	4	***1***

Algorithm 3: Local_Search. Algorithm for the local search.

Input: Sol_0 or Sol_p; `/* initial solution or perturbed solution generated with algorithm Schedule_Scenarios */`
$Sol \leftarrow Sol_0$ or $Sol \leftarrow Sol_p$;
$Sol^* \leftarrow Sol$; `/* Make` Sol `the best solution found */`
for $i = 1$ *to* $i = I$ **do**
  Change mode m randomly of activity i in solution Sol;
  Reschedule activities from i to $I + 1$ to obtain the new solution Sol' ; `/* using algorithm Schedule_Scenarios */`
  if Sol' *is feasible and* Sol' *is better than* Sol^* **then**
    $Sol^* \leftarrow Sol'$;
    $i^* \leftarrow i$;
  $i = i + 1$;
Output: Sol^*, i^*; `/* Best solution found in the neighborhood, activity` i `whose mode was changed to obtain` Sol^* `*/`

This local search delivers good local optima on short-range neighborhoods, favoring intensification over diversification. The diversification of the search is achieved with the perturbation phase and the restart strategy.

2.2.4. Perturbation and Restart

When the Local_Search algorithm (Algorithm 3) produces a local optimum, a perturbation move is applied to this local optimum. In the perturbation, the activity whose mode was changed in the local search is maintained. Only a certain number of the other activities randomly change their modes (number of movements). In this way, the part of the solution that produced a benefit is conserved in the new solution. This memory condition applies only to the perturbation phase, avoiding a bias that could produce an entrapment in a local optimum, and enforcing diversification. If this new solution is infeasible, a new perturbation is applied until a feasible solution is found (see Algorithm 4, Perturbation algorithm). This solution is used for the next iteration of the local search–perturbation cycle until a stopping criterion is reached.

The best solution found is stored when the stopping criterion is reached. After that, a complete ILS restarts, creating a new random initial solution. A certain number of restarts occur, and the best solution to be found in all restarts is returned.

Algorithm 4: Perturbation. Algorithm for the perturbation process

Input: Sol^*, i^*; /* Best solution found in the Local_Search algorithm, an activity for whose mode was changed to obtain Sol^* in the Local_Search algorithm */
$Sol \leftarrow Sol^*$;
$Sol^{**} \leftarrow Sol$; /* Make Sol the best solution found */
$i_{fixed} \leftarrow i^*$;
for $ITER = 1$ *to* $ITER = Lim\_ITER$; /* for a certain number of iterations */
do
  repeat
    for $nm = 1$ *to* $nm = Mv$; /* for a certain number of movements */
    do
      Select randomly an activity i from solution Sol, except i_{fixed};
      Select randomly a mode m for activity i;
      $nm = nm + 1$;
  until $\sum U_{im(nr)} \leq K_{nr} \quad \forall nr \in NON$; /* Non-renewable resource constraints satisfied */
  Schedule activities to obtain the perturbed solution Sol_p; /* using algorithm Schedule_Scenarios */
  Perform Local_Search algorithm on Sol_p to obtain a new solution Sol';
  if Sol' *is better than* Sol^{**} **then**
    $Sol^{**} \leftarrow Sol'$
  $Sol \leftarrow Sol'$;
  $ITER = ITER + 1$;
Output: Sol^{**}; /* Best solution found in the ILS */

2.2.5. Parameters and Assumptions of the Algorithm

A multi-start iterated local search (MS-ILS) metaheuristic proposed initially by Ramos et al. [8] for the multi-mode resource-constrained project scheduling problem (MRCPSP) was adapted in this study to solve the proposed stochastic formulation of the problem with uncertain activity duration, with the next features:

- Initial solution: The modes are assigned randomly using an adaptive heuristic procedure to produce a feasible start solution. A serial schedule generation scheme (SGS) is used.
- Feasibility handling: Infeasible solutions are not allowed.
- Local search: All of the activities are subject to the change of their mode. The modes to be changed are one at a time for each activity.
- Perturbation phase: the activity whose mode was changed to obtain the local optimum during the local search algorithm keeps its mode. Some of the other activities randomly change their modes.

The MS-ILS functioning depends on the following parameters:

- Number of iterations for the local search–perturbation cycle.
- Number of restarts.

2.3. Computational Tests

In this section, the computational resources used for the experimentation are mentioned, then the generation of the test instances is described, then the method used to calibrate the parameters of the algorithm is explained, and finally, the different experiments performed are described.

2.3.1. Computational Resources

A computer with 16GB of RAM, an Intel core i7 processor, and a 2.9 GHz CPU was used to carry out the computational tests. To code the MS-ILS algorithm, Visual C++ 2019 was used as a programming language and Visual Studio 16.3.6 as a compiler. The mathematical models of the deterministic and stochastic versions of the MRCPSP were coded using the AMPL modeling language. The solver Gurobi was used for experiments involving solving instances with an exact method. The statistical analysis software Minitab 2019 was used to perform the required statistical analyses.

2.3.2. Test Instances

Given the lack of available libraries of benchmark instances for the proposed stochastic MRCPSP, the necessary instances for these experiments were generated during this research. We adapted benchmark instances of the deterministic MRCPSP from the MMLIB50 library [59] and the J10MM, J20MM, and J30MM datasets from the PSPLIB library [60]. In this case, instances up to 50 activities were selected, because given the nature of the algorithm, they will present a computational complexity that compromises the performance of our computational equipment, particularly considering that as the number of scenarios increases, the number of iterations to execute will exponentially increase.

For these instances, the original (deterministic) parameter d_{im} (duration of activity i in mode m) was modified into a stochastic parameter $d_{im(k)}$ that follows a discrete triangular distribution with K possible realizations, each one with an associated probability $P(k)$ of occurrence. This distribution has been considered since it has been widely implemented for modeling the duration time for these types of activities (as mentioned in Section 1.1). A value of $K = 3$ for all activities and the density function displayed in Table 4 were used to create the test instances.

Table 4. Probability distribution of $d_{im(k)}$.

k	$d_{im(k)}$	Probability $P(k)$	Cumulative Probability
1	1, if $d_{im} = 1$ $d_{im} - 1$, if $d_{im} > 1$	0.25	0.25
2	d_{im}	0.50	0.75
3	$d_{im} + 1$	0.25	1.00

According to this discrete triangular distribution, the duration of each activity can take three possible values in each mode; the most likely one, with a probability of 0.50, is the value of the original parameter from the deterministic version. It can also take two other values: the value of the original parameter from the deterministic version plus 1 unit of time and the value of the original parameter from the deterministic version minus 1 unit of time, each with a probability of 0.25. In the case for when the value from the original deterministic version was 1, the value of the original deterministic version minus 1 unit would be 0; however, since the duration of an activity cannot be 0, in that case, it will take a value of 1. The reasoning behind selecting these lower and upper values lies in the fact that in the original instances, the values for d_{im} are small. Therefore, a deviation of ± 1 unit could represent relative variations up to 33% with respect to the most likely value. It is worth mentioning that all of the instances from the mentioned public libraries use only discrete values for d_{im}.

The values of k are considered to be independent for each activity i and mode m, meaning that for an instance with I activities and M modes, there are $S = K^{IM}$ possible scenarios. For example, given the proposed value of $k = 3$, an instance with 10 activities and 3 modes has 2.06×10^{14} possible scenarios. Given this large amount of possible scenarios, even for small instances, samples of scenarios were created for the test instances.

To create a sample of a certain number of scenarios for each stochastic instance, a Monte Carlo simulation procedure was carried out, generating random numbers to determine the different values of $d_{im(k)}$ according to the described probability distribution. Once each

particular scenario s is generated with a realization for each variable $d_{im(k)}$, with the Monte Carlo procedure, the duration of each activity in each mode for that particular scenario is denoted as d_{ims}. When solving an instance for a certain number of scenarios, all scenarios were considered to have the same probability of occurrence.

2.3.3. Parameter Tuning

To calibrate the number of restarts *Rs* and the number of iterations *It* of the metaheuristic algorithm to use in the experiments for the proposed stochastic MRCPSP, a pre-experiment was carried out. Twenty instances with 30 activities from the J30MM dataset of the PSPLIB library, and 20 instances with 50 activities from the MMLIB50 library were adapted to generate their stochastic versions with 20 scenarios according to the procedure described in Section 2.3.2. They were solved with the proposed MS-ILS algorithm with two different configurations: 50 iterations and 3 restarts, and 100 iterations and 10 restarts. The objective function value *O.F.* of the best solution found, and the computational execution time *ExecTime* in seconds was obtained. The results of this pre-experiment are shown in Tables 5 and 6, where *Diff. O.F* is the difference in the value of the objective function obtained with the two different configurations, and *Diff ExecTime* is the difference in the computational execution time in seconds obtained with the two different configurations.

Table 5. Results of the pre-experiment for instances with 30 activities.

	***It* = 50, *Rs* = 3**		***It* = 100, *Rs* = 10**			
Instance	***O.F.***	***ExecTime***	***O.F.***	***ExecTime***	***Diff. O.F.***	***Diff. ExecTime***
j3010_3s20	24.65	4.962	24.65	34.583	0.0%	597.0%
j3012_9s20	23.20	4.879	23.20	31.471	0.0%	545.0%
j3015_5s20	30.60	4.815	29.10	32.022	−4.9%	565.0%
j3016_5s20	31.20	5.458	31.10	33.973	−0.3%	522.4%
j3018_6s20	24.80	7.865	24.80	42.607	0.0%	441.7%
j3020_4s20	34.75	8.197	34.75	42.872	0.0%	423.0%
j3021_4s20	41.30	11.158	41.20	69.758	−0.2%	525.2%
j3022_5s20	33.70	10.190	33.70	65.115	0.0%	539.0%
j3023_10s20	22.50	7.981	22.50	33.60	0.0%	321.0%
j3025_2s20	34.65	10.144	34.65	61.084	0.0%	502.2%
				Average	−0.5%	498.2%

Table 6. Results of the pre-experiment for instances with 50 activities.

	***It* = 50, *Rs* = 3**		***It* = 100, *Rs* = 10**			
Instance	***O.F.***	***ExecTime***	***O.F.***	***ExecTime***	***Diff. O.F.***	***Diff. ExecTime***
j502_2s20	29.50	72.605	29.10	466.424	−1.4%	542.4%
j507_2s20	49.40	46.384	49.40	296.528	0.0%	539.3%
j5010_3s20	32.35	68.856	31.85	438.858	−1.5%	537.4%
j5017_3s20	17.40	36.239	17.40	230.484	0.0%	536.0%
j5025_4s20	30.35	28.432	30.10	154.289	−0.8%	442.7%
j5027_3s20	18.40	31.662	18.35	191.870	−0.3%	506.0%
j5028_5s20	21.30	34.928	21.35	222.153	0.2%	536.0%
j5039_3s20	30.25	37.155	29.55	247.680	−2.3%	566.6%
j5042_5s20	32.55	38.462	32.50	252.242	−0.2%	555.8%
j5045_3s20	44.35	59.220	44.40	376.401	0.1%	535.6%
				Average	−0.6%	529.8%

As can be seen in Tables 5 and 6, increasing the number of iterations from 50 to 100 and the number of restarts from 3 to 10 does not result in a significant difference in the value of the objective function, and thus, the quality of the solution obtained. In the case of instances with 30 activities, an average reduction of 0.5% was observed, and in the case of the instances with 50 activities, the reduction was 0.6%. On the other hand, the impact of

the change in the parameters in the computational execution time is considerable. The time to solve instances with 30 activities increased to 498.2%, and the time to solve instances with 50 activities increased to 529.8%.

These results show that 50 iterations and 3 restarts are enough to obtain good quality solutions in a reasonable period of time. Increasing the values of those parameters does not result in significantly better solutions, and results in a considerable increase in the required computing time, which is a disadvantage when solving larger instances with several scenarios. On the other hand, reducing the values of the parameters was not deemed necessary, since the computing time is already short and the quality of the solutions could be reduced. Thus, 50 iterations and 3 restarts were the parameters chosen for the rest of the experiments of the stochastic version of the problem.

2.3.4. Computational Experiments

To test the proposed multi-start iterated local search metaheuristic method, and its capability to solve the proposed stochastic multi-mode resource-constrained project scheduling problem (MRCPSP), several computational experiments were carried out once the parameters of the MS-ILS algorithm were defined.

Experiment 1. This experiment was conducted to evaluate if the proposed method to generate scenarios provides 20 scenarios that are representative of the universe of all possible scenarios. One instance with 20 activities from the J20MM dataset of the PSPLIB library was adapted to generate its stochastic version according to the procedure described in Section 2.3.2. Ten different sets of 20 scenarios were created and solved with the MS-ILS metaheuristic method, and the objective function values obtained for each set were compared.

Experiment 2. A stochasticity test was carried out to assess if solving the stochastic problem becomes relevant, compared with solving it only for the average scenario as a deterministic problem (which would be faster). A commonly accepted approach to evaluate the benefits of employing a two-stage stochastic programming model, instead of employing an equivalent deterministic version, consists of optimizing a deterministic model that assumes only a single scenario at the second stage (e.g., the average scenario), and then evaluating the latter solution with the two-stage stochastic model. This comparison allows for determining the benefits lost (cost increment) of discarding all stochastic information of the problem [63,64]. In this paper, the deterministic version consists of considering a single scenario at the second stage, which assumes the average values of processing times for each activity and at each operation mode. Then, this solution is evaluated with the stochastic version.

One instance with 20 activities from the J20MM dataset of the PSPLIB library was adapted to generate its stochastic version according to the procedure described in Section 2.3.2. Ten different sets of 20 scenarios were created and solved as stochastic problems with the MS-ILS metaheuristic method (part A). The average scenario for each of the 20 sets was also solved as a deterministic problem (part B). Then, for each set, the solution (selection of activity modes) obtained by solving the deterministic average scenario (part B) was evaluated in the 20 scenarios, and the resulting objective function value was compared with the one obtained by solving the stochastic case for the 20 scenarios (part A).

Experiment 3. To test the performance of the proposed MS-ILS metaheuristic method to solve the proposed stochastic MRCPSP, 5 instances with 10 activities and 5 instances with 20 activities from the J10MM and the J20MM datasets of the PSPLIB library were randomly selected and adapted according to the procedure described in Section 2.3.2, with 20 scenarios generated for each one. Each instance was solved using the proposed metaheuristic method and also with an exact linear programming method (LP) using Gurobi as the solver and AMPL as a programming language. The value of the objective function $O.F.$ and the computational execution time $ExecTime$ were obtained and compared. A time limit of 5400 s was set for the exact method and once reached, the solver would stop and provide the best feasible solution found, if any, in that amount of time. A comparison

with other proposed methods for solving stochastic versions of the MRCPSP was not feasible, since every author proposed different versions of the problem, with different stochastic parameters, different probability distributions and different project objectives. However, our method was compared with several other methods in Ramos et al. [8] for solving the standard deterministic version of the problem, showing good results, especially for larger instances.

Experiment 4. To evaluate whether creating more than 100 scenarios for each instance is convenient, 5 instances with 10 activities, 5 instances with 20 activities, and 5 instances with 30 activities from the J10MM, J20MM, and J30MM datasets of the PSPLIB library were selected and adapted according to the procedure described in Section 2.3.2. Each one was solved for 100 scenarios, 500 scenarios, and 1000 scenarios, and the differences in the values of the objective function $O.F.$ and in the computational execution time $ExecTime$ were recorded. Then, for each instance, the solution (selection of activity modes) obtained by solving for the 100 scenarios was evaluated in the 500 scenarios and in the 1000 scenarios, and the resulting objective function values were compared with the ones obtained by solving for the 500 scenarios and 1000 scenarios.

The results of all these experiments are presented in Section 3.

3. Results and Discussion

This section exhibits and discusses the results of the experiments described in Section 2.3.4 for testing the multi-start iterated local search metaheuristic (MS-ILS) algorithm to solve the proposed stochastic version of the multi-mode resource-constrained project scheduling problem (MRCPSP) with uncertain activity duration.

The results of the experiment to test whether the proposed method to generate scenarios provides 20 scenarios that are representative of the universe of all possible scenarios (experiment 1) are shown in Table 7, where $O.F.20S$ is the value of the objective function obtained by solving for each set of 20 scenarios. If there is little dispersion among the objective function values of the different sets, it would mean that the method is able to generate representative sets of 20 scenarios.

Table 7. Experiment 1 results. Generation of scenarios.

Instance and Set	$O.F.$ 20S
J2022_9 Set 1	23.35
J2022_9 Set 2	23.80
J2022_9 Set 3	23.55
J2022_9 Set 4	23.25
J2022_9 Set 5	23.15
J2022_9 Set 6	23.35
J2022_9 Set 7	23.21
J2022_9 Set 8	23.50
J2022_9 Set 9	23.05
J2022_9 Set 10	23.60
Mean	23.38
Variance	0.053
Standard deviation	0.230
Coefficient of variation	0.010

The values of the objective functions for each of the 10 sets of 20 scenarios are very similar, with a mean value of 23.38, a variance of 0.053, and a standard deviation of 0.230. The coefficient of variation is 0.01, which means that there is little dispersion between the different values. This confirms that each of the sets of 20 scenarios is very similar to the others, and thus suggests that they are a representative sample of the universe of possible scenarios.

The results of the stochasticity experiment (experiment 2) to test if solving the stochastic problem becomes relevant, compared with solving the deterministic problem only for the average scenario, are shown in Table 8, where $O.F.\ 20S$ is the value of the objective function obtained by solving for each set of 20 scenarios (part A), $O.F.\ Avg.\ Eval.\ 20S$ is the objective function value obtained by taking the solution (activity modes) of the deterministic problem corresponding to the average scenario (part B), and evaluating it in the 20 scenarios, and *Diff.* stands for the percentage of the difference between both objective function values. If the objective function value from solving the stochastic problem for the 20 scenarios is lower than the value of the objective function obtained by solving the deterministic problem for the average scenario and evaluating it for the 20 scenarios, it would mean that solving the stochastic model is relevant.

Table 8. Experiment 2 results. Stochasticity.

Instance and Set	$O.F.\ 20S$	$O.F.\ Avg.\ Eval.\ 20S$	$Diff.$
J2022_9 Set 1	23.35	24.35	4.28%
J2022_9 Set 2	23.80	27.50	15.55%
J2022_9 Set 3	23.55	25.05	6.37%
J2022_9 Set 4	23.25	25.35	9.03%
J2022_9 Set 5	23.15	26.85	15.98%
J2022_9 Set 6	23.35	26.80	14.78%
J2022_9 Set 7	23.21	27.95	20.42%
J2022_9 Set 8	23.50	24.65	4.89%
J2022_9 Set 9	23.05	24.40	5.86%
J2022_9 Set 10	23.60	27.05	14.62%
Average			11.188%

Comparing the value of the objective function from solving the stochastic problem for the 20 scenarios with the objective function obtained by evaluating the solution of the average scenario in the 20 scenarios, for each of the 10 sets, it can be seen in Table 8 that the value of the objective function obtained by solving the stochastic problem is consistently lower than the value of the objective function obtained by solving the deterministic problem for the average scenario and evaluating it for the 20 scenarios. The difference is significant, 11.18% on average, which justifies the need to solve the stochastic problem for a set of scenarios instead of solving the deterministic problem for the average scenario.

The results of experiment 3 to test the performance of the proposed MS-ILS metaheuristic method to solve the proposed stochastic MRCPSP, by comparing it to the performance of an exact linear programming method using Gurobi as solver and AMPL as programming language are shown in Tables 9 and 10, where $O.F.$ is the value of the objective function, $ExecTime$ is the computational execution time in seconds, *Diff. O.F.* is the difference in the value of the objective function obtained with the two methods, *Diff. ExecTime* is the difference in the computational execution time in seconds with the two methods, $Optimal$ shows whether the solution from the exact method is optimal, and N/F means that no feasible solution was found within the time limit.

Table 9. Experiment 3 results. Performance test using instances with 10 activities.

	MS-ILS Metaheuristic		AMPL/Gurobi Exact				
Instance	***O.F.***	***ExecTime***	***O.F.***	***ExecTime***	***Optimal***	***Diff. O.F.***	***Diff. ExecTime***
J1019_5	13.20	0.716	13.20	256	Yes	0.00%	35,678%
J1023_6	19.60	0.985	19.60	2696	Yes	0.00%	273,631%
J1028_4	15.95	0.795	15.95	203	Yes	0.00%	25,488%
J1030_9	16.60	2.080	16.00	548	Yes	−3.61%	26,258%
J1035_1	29.35	1.429	N/F	5400			
Average		1.201				−0.91%	90,264%

Table 10. Experiment 3 results. Performance test using instances with 20 activities.

	MS-ILS Metaheuristic		AMPL/Gurobi Exact				
Instance	***O.F.***	***ExecTime***	***O.F.***	***ExecTime***	***Optimal***	***Diff. O.F.***	***Diff. ExecTime***
J2013_5	39.70	5.484	N/F	5400			
J2015_6	21.35	3.633	N/F	5400			
J2018_1	30.15	4.756	30.15	867	Yes	0.00%	18,147%
J2020_3	21.85	4.421	21.85	631	Yes	0.00%	14,192%
J2022_9	23.35	3.718	22.95	4138	Yes	−1.71%	111,221%
Average		4.402				−0.57%	47,853%

Tables 9 and 10 show that while the proposed MS-ILS algorithm solved all of the instances without problems in a short amount of time (1.201 s on average for the instances with 10 activities and 4.402 s on average for the instances with 20 activities), the exact method struggles to solve them. One instance with 10 activities and 2 instances with 20 activities could not be solved with the exact method within the time limit of 5400 s. For the instances that could be solved using both methods, the exact method took on average 90,264% more time than the metaheuristic algorithm for the instances with 10 activities and 47,853% more time for the instances with 20 activities. It was expected that the metaheuristic would consume less time than the exact method, but it is relevant to report how much the difference was. The values of the objective function of the solutions found with the metaheuristic method are very similar to those found with the exact method. On average, the exact method found values less than 1% lower than the metaheuristic, and in several of the instances, both methods found the same value, which is optimal. This shows that the proposed MS-ILS metaheuristic method provides good quality solutions in a small fraction of the time elapsed using the exact method.

The results of experiment 4, to evaluate if creating more than 100 scenarios for each instance is convenient, are shown in Tables 11–17. If there is not a significant difference between the objective function value obtained by solving for 100 scenarios and the objective function value obtained by evaluating for a greater number of scenarios (500 or 1000) the solution obtained by solving for 100 scenarios, it would mean that using more than 100 scenarios is not necessary.

Tables 11–13 show the objective function value ($O.F.$) and the execution time ($ExecTime$) obtained when solving the stochastic problem for 100 scenarios, 500 scenarios, and 1000 scenarios using instances with 10 project activities, 20 project activities, and 30 project activities, respectively. The comparisons between those results are shown in Table 14, where $Avg.\ Diff.\ 500S - 100S$ is the average percentage of the difference between 500 scenarios and 100 scenarios, and $Avg.\ Diff.\ 1000S - 100S$ is the average percentage of the difference between 1000 scenarios and 100 scenarios. Tables 15–17 show the results of comparing the objective function values obtained by solving for 500 scenarios ($O.F.\ 500S$) versus by evaluating for the 500 scenarios the solution obtained by solving for 100 scenarios ($O.F.100SEval500S$) and comparing the objective function values obtained by solving for 1000 scenarios ($O.F.\ 1000S$), versus by evaluating for the 1000 scenarios the solution ob-

tained by solving for 100 scenarios (*O.F.* 100*S Eval* 1000*S*), for the 3 different sizes of instances (10 activities, 20 activities, and 30 activities).

Table 11. Experiment 4 results. Different number of scenarios. Instances with 10 activities.

	100 Scenarios		**500 Scenarios**		**1000 Scenarios**	
Instance	***O.F.***	***ExecTime***	***O.F.***	***ExecTime***	***O.F.***	***ExecTime***
J1019_5	13.18	2.665	13.25	12.339	13.32	27.151
J1023_6	19.76	3.005	19.75	14.700	19.76	40.996
J1028_4	16.08	2.421	16.06	11.478	16.07	21.239
J1030_9	16.30	2.942	15.94	12.652	16.35	28.738
J1035_1	29.01	4.481	29.06	21.853	29.12	42.621

Table 12. Experiment 4 results. Different number of scenarios. Instances with 20 activities.

	100 Scenarios		**500 Scenarios**		**1000 Scenarios**	
Instance	***O.F.***	***ExecTime***	***O.F.***	***ExecTime***	***O.F.***	***ExecTime***
J2013_5	39.68	24.936	39.48	121.946	39.52	255.132
J2015_6	21.69	13.747	21.41	66.311	21.79	131.782
J2018_1	30.68	17.547	30.38	90.377	30.39	171.864
J2020_3	21.53	15.395	21.58	65.655	21.69	132.923
J2022_9	23.66	14.333	23.52	68.464	23.51	137.617

Table 13. Experiment 4 results. Different number of scenarios. Instances with 30 activities.

	100 Scenarios		**500 Scenarios**		**1000 Scenarios**	
Instance	***O.F.***	***ExecTime***	***O.F.***	***ExecTime***	***O.F.***	***ExecTime***
j3012_9	22.83	44.290	22.97	195.676	22.92	388.492
j3016_5	31.38	43.619	31.47	197.785	31.44	494.349
j3020_4	35.31	59.833	35.41	292.149	35.49	605.811
j3022_5	34.05	93.276	33.92	457.841	33.97	831.533
j3025_2	34.81	73.723	35.00	364.050	35.05	755.391

Table 14. Experiment 4 results. Comparisons between the results obtained considering different numbers of scenarios.

	$Avg.\,Diff.$ **500***S* − **100***S*		$Avg.\,Diff.$ **1000***S* − **100***S*	
Instances	***O.F.***	***ExecTime***	***O.F.***	***ExecTime***
With 10 activities	−0.32%	368%	0.33%	937%
With 20 activities	−0.62%	378%	−0.16%	856%
With 30 activities	0.27%	373%	0.30%	887%

As can be seen in Table 14, using more scenarios does not provide a significant difference in the value of the objective function of the best solution found. Changing from 100 scenarios to 500 scenarios resulted in a difference of −0.32% for the instances with 10 activities, −0.62% for the instances with 20 activities and 0.27% for instances with 30 activities. Changing from 100 scenarios to 1000 scenarios resulted in a difference of 0.33% for the instances with 10 activities, −0.16% for the instances with 20 activities, and 0.30% for instances with 30 activities. All of those differences are significantly small.

In terms of the computational execution time, it is clear that there is a direct relationship between the number of scenarios and the time taken to reach the solution. Changing from 100 scenarios to 500 scenarios resulted in a difference of 368% in the execution time for the instances with 10 activities, 378% for the instances with 20 activities, and 373% for instances with 30 activities. Changing from 100 scenarios to 1000 scenarios resulted in a difference of 937% for the instances with 10 activities, 856% for the instances with 20 activities, and 887% for instances with 30 activities. This is shown in Table 14.

As can be observed in Tables 15–17, where $Diff.A$ is the percentage of the difference between the value of the objective function obtained by solving for 500 scenarios and the value of the objective function obtained by evaluating in the 500 scenarios, the solution (activity modes) is obtained by solving for 100 scenarios; $Diff.B$ is the percentage of the difference between the value of the objective function obtained by solving for 1000 scenarios and the value of the objective function obtained by evaluating in the 1000 scenarios the solution (activity modes) obtained by solving for 100 scenarios; those differences are significantly small. Taking the solution from solving for 100 scenarios and evaluating it in 500 scenarios resulted in objective function values of 0.53% higher for instances with 10 activities, 0.37% higher for instances with 20 activities, and 0.03% lower for instances with 30 activities, than solving them for the 500 scenarios. Taking the solution from solving for 100 scenarios and evaluating it in 1000 scenarios resulted in objective function values that were 0.01% higher for instances with 10 activities, 0.00% higher for instances with 20 activities, and 0.011% higher for instances with 30 activities, than solving them for the 1000 scenarios. This means that, for this stochastic version of the problem with the proposed probability distribution, using 100 scenarios is enough to obtain a good quality solution, and increasing the number of scenarios above 100 does not result in a substantial difference in the value of the objective function, but only increases the execution time considerably.

Table 15. Experiment 4 results. Evaluation of the 100 scenarios solution in the 500 and 1000 scenarios. Ten activities instances.

Instance	*O.F.* 500*S*	*O.F.* 100*S Eval.* 500*S*	*Diff.A*	*O.F.* 1000*S*	*O.F.* 100*S Eval.* 1000*S*	*Diff.B*
J1019_5	13.254	13.286	0.24%	13.317	13.324	0.05%
J1023_6	19.752	19.752	0.00%	19.755	19.755	0.00%
J1028_4	16.062	16.062	0.00%	16.070	16.070	0.00%
J1030_9	15.944	16.331	2.43%	16.353	16.353	0.00%
J1035_1	29.056	29.056	0.00%	29.117	29.117	0.00%
Average			0.53%			0.01%

Table 16. Experiment 4 results. Evaluation of the 100 scenarios solution in the 500 and 1000 scenarios. Twenty activities instances.

Instance	*O.F.* 500*S*	*O.F.* 100*S Eval.* 500*S*	*Diff.A*	*O.F.* 1000*S*	*O.F.* 100*S Eval.* 1000*S*	*Diff.B*
J2013_5	39.478	39.666	0.48%	39.522	39.664	0.36%
J2015_6	21.412	21.694	1.32%	21.789	21.707	−0.38%
J2018_1	30.384	30.384	0.00%	30.387	30.387	0.00%
J2020_3	21.584	21.588	0.02%	21.692	21.693	0.00%
J2022_9	23.522	23.526	0.02%	23.511	23.511	0.00%
Average			0.37%			0.00%

Table 17. Experiment 4 results. Evaluation of the 100 scenarios solution in the 500 and 1000 scenarios. Thirty activities instances.

Instance	*O.F.* 500*S*	*O.F.* 100*S Eval.* 500*S*	*Diff.A*	*O.F.* 1000*S*	*O.F.* 100*S Eval.* 1000*S*	*Diff.B*
J3012_9	22.974	22.912	−0.27%	22.916	22.916	0.00%
J3016_5	31.468	31.476	0.03%	31.436	31.436	0.43%
J3020_4	35.406	35.413	0.02%	35.488	35.488	0.02%
J3022_5	33.918	33.946	0.08%	33.972	33.972	0.11%
J3025_2	35.002	35.006	0.01%	35.048	35.048	0.00%
Average			−0.03%			0.11%

4. Conclusions

The multi-mode resource-constrained project scheduling problem (MRCPSP) is a well-known NP-hard optimization problem that has been studied by several researchers in recent history. Several heuristic and metaheuristic methods have been proposed to solve its deterministic version. However, as stated in Section 1, only four published studies were found regarding stochastic versions of the problem, each one proposing a different version with different stochastic parameters, different probability distributions, and different methods to solve them.

One contribution of this research is the formulation of a previously unstudied stochastic version of the MRCPSP with uncertain activity duration, along with a method to create test instances for that version of the problem, with its proposed probability distribution.

Another contribution of this research is the successful adaptation of a recently proposed multi-start iterated local search algorithm (MS-ILS) to solve the proposed stochastic MRCPSP. The result of the experiments show that the stochastic model is relevant, that the MS-ILS algorithm is capable and efficient in solving it compared with exact methods, that the method to generate scenarios provides a set of scenarios that is representative of the universe of scenarios, and that using 100 scenarios is enough to obtain good quality solutions in a very short time. There are very few studies regarding stochastic versions of this problem, as described in Section 1.1; thus, the importance of developing new methods to solve it, contributing to filling that gap.

The proposed algorithm, which was shown to be capable and efficient for solving the proposed stochastic formulation of the problem, can be taken as a starting point for further research. For example, the best solution found by the proposed MS-ILS algorithm could be used as an initial solution for an exact mathematical method, thus creating a matheuristic algorithm. As future research, specialized metaheuristics such as Tabu Search, Variable Neighborhood Search, or Particle Swarm Optimization can be developed to compare their performances to the one proposed here. In addition, the algorithm can be tested to solve different versions of the stochastic case, apart from the one studied in this research; for example, considering different probability distributions for the stochastic parameter or considering uncertainty in another parameter such as the activity usage of non-renewable resources, which in a real-life application could be the monetary cost.

Author Contributions: Conceptualization, A.S.R., P.A.M.-G., S.N.-G. and E.O.-B.; Data curation, A.S.R.; Formal analysis, A.S.R., P.A.M.-G., S.N.-G. and E.O.-B.; Funding acquisition, S.N.-G. and E.O.-B.; Investigation, A.S.R., P.A.M.-G. and S.N.-G.; Methodology, A.S.R., P.A.M.-G. and E.O.-B.; Project administration, E.O.-B.; Resources, A.S.R.; Software, A.S.R.; Supervision, P.A.M.-G., S.N.-G. and E.O.-B.; Validation, A.S.R., P.A.M.-G., S.N.-G. and E.O.-B.; Visualization, A.S.R.; Writing—original draft, A.S.R.; Writing—review and editing, A.S.R., P.A.M.-G., S.N.-G. and E.O.-B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Universidad Panamericana (grant number UP-CI-2022-GDL-11-ING).

Data Availability Statement: The data used for the experiments of this study are openly available in PSPLIB at http://www.om-db.wi.tum.de/psplib/getdata_mm.html (accessed on 1 September 2021) [60], and in MMLIB at https://www.projectmanagement.ugent.be/research/data (accessed on 1 September 2021) [59].

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

MRCPSP	Multi-mode resource-constrained project scheduling problem
MS-ILS	Multi-start iterated local search
RCPSP	Resource-constrained project scheduling problem
NP-hard	Non-deterministic polynomial-time hard
MILP	Mixed-integer linear programming
SGS	Schedule generation scheme

References

1. Blazewicz, J.; Lenstra, J.K.; Kan, A.H. Scheduling subject to resource constraints: Classification and complexity. *Discret. Appl. Math.* **1983**, *5*, 11–24. [CrossRef]
2. Zhang, S. Selection of Multimode Resource-Constrained Project Scheduling Scheme Based on DEA Method. *Sci. Program.* **2021**, *2020*, 1–7. [CrossRef]
3. 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. [CrossRef]
4. Zhu, G.; Bard, J.F.; Yu, G. A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. *INFORMS J. Comput.* **2006**, *18*, 283–406. [CrossRef]
5. Sprecher, A.; Hartmann, S.; Drexl, A. An exact algorithm for project scheduling with multiple modes. *Oper.-Res.-Spektrum* **1997**, *19*, 195–203. [CrossRef]
6. Sprecher, A.; Drexl, A. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. *Eur. J. Oper. Res.* **1998**, *107*, 431–450. [CrossRef]
7. Chakrabortty, R.K.; Abbasi, A.; Ryan, M.J. Multi-mode resource-constrained project scheduling using modified variable neighborhood search heuristic. *Int. Trans. Oper. Res.* **2019**, *27*, 138–167. [CrossRef]
8. 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*, 1–23. [CrossRef]
9. Talbi, E.G. *Metaheuristics*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2009. [CrossRef]
10. Hartmann, S.; Kolisch, R. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2000**, *127*, 394–407. [CrossRef]
11. Özdamar, L.; Ulusoy, G. A survey on the resource-constrained project scheduling problem. *IIE Trans.* **1995**, *27*, 574–586. [CrossRef]
12. Demeulemeester, E.L.; Herroelen, W.S. New benchmark results for the resource-constrained project scheduling problem. *Manag. Sci.* **1997**, *43*, 1469–1608. [CrossRef]
13. 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]
14. 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]
15. Elmaghraby, S.E. *Activity Networks: Project Planning and Control by Network Models*; John Wiley & Sons: New York, NY, USA, 1977.
16. Talbot, F.B. Resource-Constrained Project Scheduling with Time-Resource Tradeoffs: The Nonpreemptive Case. *Manag. Sci.* **1982**, *28*, 1091–1213. [CrossRef]
17. Kolisch, R.; Drexl, A. Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE Trans.* **1997**, *29*, 987–999. [CrossRef]
18. Mori, M.; Tseng, C.C. A genetic algorithm for multi-mode resource constrained project scheduling problem. *Eur. J. Oper. Res.* **1997**, *100*, 134–141. [CrossRef]
19. Hartmann, S. Project Scheduling with Multiple Modes: A Genetic Algorithm. *Ann. Oper. Res.* **2001**, *102*, 111–135. [CrossRef]
20. Alcaraz, J.; Maroto, C.; Ruiz, R. Solving the Multi-Mode Resource-Constrained Project Scheduling Problem with genetic algorithms. *J. Oper. Res. Soc.* **2003**, *54*, 614–626. [CrossRef]
21. Lova, A.; Tormos, P.; Cervantes, M.; Barber, F. An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *Int. J. Prod. Econ.* **2009**, *117*, 302–316. [CrossRef]
22. Van Peteghem, V.; Vanhoucke, M. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2010**, *201*, 409–418. [CrossRef]
23. Sebt, M.H.; Afshar, M.R.; Alipouri, Y. An efficient genetic algorithm for solving the multi-mode resource-constrained project scheduling problem based on random key representation. *Int. J. Supply Oper. Manag.* **2015**, *2*, 905–924. [CrossRef]
24. Zamani, R. An effective mirror-based genetic algorithm for scheduling multi-mode resource constrained projects. *Comput. Ind. Eng.* **2019**, *127*, 914–924. [CrossRef]
25. Zhang, H.; Tam, C.M.; Li, H. Multimode project scheduling based on particle swarm optimization. *Comput.-Aided Civ. Infrastruct. Eng.* **2006**, *21*, 93–103. [CrossRef]
26. Jarboui, B.; Damak, N.; Siarry, P.; Rebai, A. A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Appl. Math. Comput.* **2008**, *195*, 299–308. [CrossRef]

27. Ranjbar, M.; De Reyck, B.; Kianfar, F. A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. *Eur. J. Oper. Res.* **2009**, *193*, 35–48. [CrossRef]
28. Van Peteghem, V.; Vanhoucke, M. Using resource scarceness characteristics to solve the multi-mode resource-constrained project scheduling problem. *J. Heuristics* **2011**, *17*, 705–728. [CrossRef]
29. Damak, N.; Jarboui, B.; Siarry, P.; Loukil, T. Differential evolution for solving multi-mode resource-constrained project scheduling problems. *Comput. Oper. Res.* **2009**, *36*, 2653–2659. [CrossRef]
30. Elloumi, S.; Fortemps, P. A hybrid rank-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2010**, *205*, 31–41. [CrossRef]
31. Wang, L.; Fang, C. An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem. *Comput. Oper. Res.* **2012**, *39*, 449–460. [CrossRef]
32. Chiang, C.W.; Huang, Y.Q.; Wang, W.Y. Ant colony optimization with parameter adaptation for multi-mode resource-constrained project scheduling. *J. Intell. Fuzzy Syst.* **2008**, *19*, 345–358.
33. Li, H.; Zhang, H. Ant colony optimization-based multi-mode scheduling under renewable and nonrenewable resource constraints. *Autom. Constr.* **2013**, *35*, 431–438. [CrossRef]
34. Wauters, T.; Verbeeck, K.; Berghe, G.V.; De Causmaecker, P. Learning agents for the multi-mode project scheduling problem. *J. Oper. Res. Soc.* **2011**, *62*, 281–290. [CrossRef]
35. Słowiński, R.; Soniewicki, B.; Węglarz, J. DSS for multiobjective project scheduling. *Eur. J. Oper. Res.* **1994**, *79*, 220–229. [CrossRef]
36. Józefowska, J.; Mika, M.; Rózycki, R.; Waligóra, G.; Węglarz, J. Simulated Annealing for Multi-Mode Resource-Constrained Project Scheduling. *Ann. Oper. Res.* **2001**, *102*, 137–155. [CrossRef]
37. Bouleimen, K.; Lecocq, H. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *Eur. J. Oper. Res.* **2003**, *149*, 268–281. [CrossRef]
38. Fernandes Muritiba, A.E.; Rodrigues, C.D.; Araùjo da Costa, F. A Path-Relinking algorithm for the multi-mode resource-constrained project scheduling problem. *Comput. Oper. Res.* **2018**, *92*, 145–154. [CrossRef]
39. Chen, W.N.; Zhang, J. Scheduling multi-mode projects under uncertainty to optimize cash flows: A Monte Carlo ant colony system approach. *J. Comput. Sci. Technol.* **2012**, *27*, 950–965. [CrossRef]
40. Chakrabortty, R.K.; Ryan, M.J. Robust Optimization Based Heuristic Approach for Solving Stochastic Multi-Mode Resource Constrained Project Scheduling Problem. In Proceedings of the 2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, 14–17 December 2020; pp. 1157–1161.
41. Balouka, N.; Cohen, I. A robust optimization approach for the multi-mode resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2021**, *291*, 457–470. [CrossRef]
42. Xie, F.; Li, H.; Xu, Z. Multi-mode resource-constrained project scheduling with uncertain activity cost. *Expert Syst. Appl.* **2021**, *168*. [CrossRef]
43. Azimi, P.; Sholekar, S. A Simulation Optimization Approach for The Multi-Objective Multi-Mode Resource Constraint Project Scheduling Problem. *Int. J. Ind. Eng. Prod. Res.* **2021**, *32*, 37–45.
44. Yuan, Y.; Ye, S.; Lin, L.; Gen, M. Multi-objective multi-mode resource-constrained project scheduling with fuzzy activity durations in prefabricated building construction. *Comput. Ind. Eng.* **2021**, *158*, 107316. [CrossRef]
45. Kokonendji, C.; Senga Kiesse, T.; Zocchi, S.S. Discrete triangular distributions and non-parametric estimation for probability mass function. *J. Nonparametric Stat.* **2007**, *19*, 241–254. [CrossRef]
46. Deblaere, F.; Demeulemeester, E.; Herroelen, W. Proactive policies for the stochastic resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2011**, *214*, 308–316. [CrossRef]
47. Brčić, M.; Katić, M.; Hlupić, N. Planning horizons based proactive rescheduling for stochastic resource-constrained project scheduling problems. *Eur. J. Oper. Res.* **2019**, *273*, 58–66. [CrossRef]
48. Kesen, S.E. Capacity-constrained supplier selection model with lost sales under stochastic demand behaviour. *Neural Comput. Appl.* **2014**, *24*, 347–356. [CrossRef]
49. Urgo, M.; Váncza, J. A branch-and-bound approach for the single machine maximum lateness stochastic scheduling problem to minimize the value-at-risk. *Flex. Serv. Manuf. J.* **2019**, *31*, 472–496. [CrossRef]
50. Wen-Huei Yang, Kamlesh Mathur, R.H.B. Stochastic Vehicle Routing Problem with Restocking. *Transp. Sci.* **2000**, *34*, 99–112. [CrossRef]
51. Miranda, D.M.; Conceição, S.V. The vehicle routing problem with hard time windows and stochastic travel and service time. *Expert Syst. Appl.* **2016**, *64*, 104–116. [CrossRef]
52. Michallet, J.; Prins, C.; Amodeo, L.; Yalaoui, F.; Vitry, G. Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services. *Comput. Oper. Res.* **2014**, *41*, 196–207. [CrossRef]
53. Nguyen, V.P.; Prins, C.; Prodhon, C. A multi-start iterated local search with tabu list and path relinking for the two-echelon location-routing problem. *Eng. Appl. Artif. Intell.* **2012**, *25*, 56–71. [CrossRef]
54. Avci, M.; Topaloglu, S. A multi-start iterated local search algorithm for the generalized quadratic multiple knapsack problem. *Comput. Oper. Res.* **2017**, *83*, 54–65. [CrossRef]

55. Sassi, O.; Cherif-Khettaf, W.R.; Oulamara, A. Multi-start Iterated Local Search for the Mixed Fleet Vehicle Routing Problem with Heterogenous Electric Vehicles. In Proceedings of the 15th European Conference, Evolutionary Computation in Combinatorial Optimization, openhagen, Denmark, 8–10 April 2015; Ochoa, G., Chicano, F., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 138–149. [CrossRef]
56. Venkatesh, P.; Srivastava, G.; Singh, A. A Multi-start Iterated Local Search Algorithm with Variable Degree of Perturbation for the Covering Salesman Problem. In *Proceedings of the Harmony Search and Nature Inspired Optimization Algorithms. Advances in Intelligent Systems and Computing*; Yadav, N., Yadav, A., Bansal, J.C., Deep, K., Kim, J.H., Eds.; Springer: Singapore, 2019; Volume 741, pp. 279–292. [CrossRef]
57. Guan, J.; Lin, G.; Feng, H.B. A multi-start iterated local search algorithm for the uncapacitated single allocation hub location problem. *Appl. Soft Comput.* **2018**, *73*, 230–241. [CrossRef]
58. Gokalp, O.; Ugur, A. A multi-start ILS–RVND algorithm with adaptive solution acceptance for the CVRP. *Soft Comput.* **2020**, *24*, 2941–2953. [CrossRef]
59. Van Peteghem, V.; Vanhoucke, M. An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *Eur. J. Oper. Res.* **2014**, *235*, 62–72. [CrossRef]
60. 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]
61. Ballestín, F. When it is worthwhile to work with the stochastic RCPSP? *J. Sched.* **2007**, *10*, 153–166. [CrossRef]
62. Sprecher, A.; Kolisch, R.; Drexl, A. Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **1995**, *80*, 94–102. [CrossRef]
63. Alfieri, A.; Tolio, T.; Urgo, M. A two-stage stochastic programming project scheduling approach to production planning. *Int. J. Adv. Manuf. Technol.* **2012**, *62*, 279–290. [CrossRef]
64. Paul, J.A.; Zhang, M. Supply location and transportation planning for hurricanes: A two-stage stochastic programming framework. *Eur. J. Oper. Res.* **2019**, *274*, 108–125. [CrossRef]