Asia-Pacific Journal of Operational Research
Vol. 41, No. 1 (2024) 2350008 (29 pages)

DOI: 10.1142/S0217595923500082

Minimizing the Expected Renewable Resource Costs in a Project with Stochastic Resource Availability

Hossein Moghaddaszadeh
Department of Industrial Engineering
Faculty of Engineering, Ferdowsi
University of Mashhad, Iran

Mohammad Ranjbar*
Department of Industrial Engineering
Faculty of Engineering, Ferdowsi
University of Mashhad, Iran

Negin Jamili
Rotterdam School of Management
Erasmus University, Netherlands

Received 15 January 2021
Revised 23 November 2021
Accepted 6 February 2022
Published 27 May 2023

In this paper, we study the well-known resource availability cost problem with stochastic resource availability. The objective is to determine the initial levels of all renewable resources and establish a schedule corresponding to each scenario such that the expected resource availability cost is minimized. We assume that resource shortfalls can be compensated externally but at a noticeable higher cost. We formulate the problem as a two-stage stochastic programming model (TSSPM). We also develop an exact decomposition-based algorithm (DBA) for the particular case of the problem with at most two resources, which also functions as a heuristic for the original problem. Since the number of scenarios influences the performance of the developed solution approaches, we utilize a fast scenario reduction method to reduce the number of scenarios. Computational results indicate that the DBA outperforms the TSSPM formulation in solution quality and CPU runtime.

Keywords: Project scheduling; resource availability cost problem; two-stage stochastic programming.

1. Introduction

Resource availability has been one of the main challenges in project scheduling since it directly impacts the progress of projects. Any disruption caused by resource

*Corresponding author.

unavailability or failure of resources may lead to high costs and missed deadlines. Therefore, dealing with the costs of resource availability is an important task, which was introduced by Möhring (1984) as the *resource availability cost problem* (RACP). This problem has been formulated to minimize the total cost of unlimited renewable resources required to complete the project before a pre-specified deadline.

In construction, oil, and gas projects, various renewable resources such as concrete mixer machines, foundation drilling machines, mining machines, crushing machines, cranes, and loaders might be utilized where each of these resources may need emergency maintenance operations after breakdowns. Emergency maintenance usually happens at stochastic times and takes a stochastic duration which can be modeled by a set of scenarios based on reliability models (Jardine and Tsang, 2013). These models employ age, type, quantity, and quality of machine utilization to predict the time and duration of stochastic breakdowns (Bei and Zhu, 2017). The stochastic duration of emergency maintenance operations causes uncertain disruptions in renewable resources and highly likely leads to higher costs due to missed due dates and deadlines, resource idleness, and higher work-in-process inventory (Herroelen and Leus, 2005). As a result, it is critically important to address these uncertainties. Moreover, many supply chains might encounter serious challenges and disruptions such that rarely orders are received on time. Consequently, we assume that renewable resources must have been booked a few times ago before the project's start. These realistic assumptions give rise to the definition of a stochastic variant of the RACP, which is referred to as the *resource availability cost problem with stochastic resource availability* (RACPSRA).

The contributions of this paper are three-fold: (1) we extend the formulation of Möhring (1984) to model the RACPSRA as a *two-stage stochastic programming model* (TSSPM) formulation; (2) we develop a *decomposition-based algorithm* (DBA); and finally (3) we show that our proposed DBA solves the problem to optimally when the number of renewable resources is not larger than two. This paper is arranged as follows. Section 2 provides a literature review on the subject. In Sec. 3, we describe our problem and propose a TSSPM formulation. Then in Sec. 4, we develop our DBA and discuss scenario generation and reduction methods in Sec. 5. Comparative computational results are discussed in Sec. 6. Finally, conclusions and future research directions are provided in Sec. 7.

2. Literature Review

The *resource-constrained project scheduling problem* (RCPSP) is one of the most studied problems in the field of project scheduling, for which there is a vast body of literature. We refer interested readers to the papers by Habibi *et al.* (2018) and Hartmann and Briskorn (2010), who surveyed the recent developments of the RCPSP.

The RACP is closely related to the RCPSP and was firstly studied by Möhring (1984), who showed the NP-hardness of the problem. A few years later,

Demeulemeester (1995) presented an efficient optimization method based on a previously developed branch-and-bound procedure for the RCPSP. Afterward, Rodrigues and Yamashita (2010) introduced a new exact approach that significantly improved the algorithm designed by Demeulemeester (1995). In a more recent study, Coughlan *et al.* (2015) considered the multi-mode variant of the RCPSP in which resource calendars are respected, and developed an exact state-of-the-art branch-price-and-cut algorithm. Finally, considering calendar constraints, Kreter *et al.* (2018) studied the RACP/max, the RACP with maximum and minimum time lags, and formulated the problem as a mixed-integer linear program, and presented a constraint programming approach.

In addition to the mentioned exact approaches, some authors have implemented metaheuristic methods to solve the RACP. As an example, Hsu and Kim (2005) presented a new heuristic for the multi-mode resource investment problem, originated from the RCPSP with simultaneous due dates and resource constraints. A similar problem was studied by Shadrokh and Kianfar (2007), and they permitted tardiness of the project with a defined penalty and solved the problem using a genetic algorithm. We also cite Yamashita *et al.* (2006) who implemented several variants of scatter search for the RACP, Ranjbar and Shadrokh (2008) who tackled the problem by developing two meta-heuristics, namely a path relinking algorithm and a genetic algorithm, and finally, Van Peteghem and Vanhoucke (2013) who presented an artificial immune system algorithm for the RACP.

There are many papers investigated other variants of the RACP. Afshar-Nadjafi (2014) presented a mixed integer programming formulation and a simulated annealing algorithm for the multi-mode RACP with recruitment constraints and release dates for resources. The multi-mode version of resources was also studied by Qi *et al.* (2015) who designed a schedule generation scheme and proposed a particle swarm optimization algorithm for the problem. Finally, in a more recent study, Van Peteghem and Vanhoucke (2015) presented an invasive weed optimization algorithm to solve a RACP that minimizes total cost and lateness minimization.

A prevalent trend in the literature is studying project scheduling under stochasticity or uncertainty. The stochastic RCPSP has received considerable attention during the last two decades. Lambrechts *et al.* (2008a) modeled RCPSP with uncertain resource availability subject to unforeseen breakdowns and proposed three reactive policies to deal with disruptions. They extended their work and aimed to generate robust baseline schedules using a procedure based upon a tabu search algorithm (Lambrechts *et al.*, 2008b). After that, they analytically investigated the impact of unexpected resource breakdowns on the duration of the activities (Lambrechts *et al.*, 2011). Furthermore, Ji and Yao (2017) investigated a multi-objective project scheduling problem under both uncertain duration and uncertain resource allocation times of activities. Moreover, Fu *et al.* (2015) studied a resource-constrained project scheduling problem with minimum and maximum time lags (RCPSP/max), stochastic duration, and unreliable resources. Li and Womer (2015) focused on tackling large size instances of a stochastic RCPSP and developed efficient approximate

dynamic programming algorithms. Recently, Bruni *et al.* (2017) developed a general decomposition approach for a robust RCPSP and proposed an adaptive robust optimization approach, and Uysal *et al.* (2018) studied stochastic resource demand possibilities in the RCPSP and utilized chance constraints to develop a piece-wise linear mixed-integer programming formulation. Moreover, Nabipoor Afruzi *et al.* (2018) investigated renewable resource constraints in a weighted multi-project scheduling problem with stochastic duration, and Bruni *et al.* (2018) proposed a two-stage stochastic programming formulation for the RCPSP under uncertainty.

Although there is extensive literature on stochastic RCPSP, there are few stochastic variants of the RACP. Yamashita *et al.* (2007) modeled a RACP with an uncertain duration of activities by having the uncertainty be supported by a set of scenarios. To explore robust solutions regarding all scenarios, they developed a robust optimization procedure. They created two models, one of which aimed to minimize a maximum regret function, whereas the other one targeted the minimization of a mean-variance function. Furthermore, Moradi *et al.* (2019) considered the project scheduling problem under uncertain resource availability and activity duration. They assumed a coalition among Subcontractors of the construction projects to achieve their goals, i.e., minimizing the makespan and maximizing the resource availability and their profits. They developed a robust optimization model for the RCPSP considering the three objectives in which the uncertainty is modeled in a set of scenarios. Chakrabortty *et al.* (2019) investigated the influence of uncertainty in activity duration and resource availability on project time and cost. They came up with a risk assessment framework to model uncertainty in the problem. They developed a heuristic approach, titled enhanced move-based local search, to solve the problem.

3. Problem Description and Modeling

We consider a project defined by a set $N = \{0, 1, \ldots, n+1\}$ of activities where activities 0 and $n+1$ indicate dummy activities representing the start and the end of the project, respectively. All activities must be finished no later than a given project's deadline (δ). The project is characterized with a precedence graph $G = (N, A)$ with $A \subset \{(i, j) \,|\, i, j \in N, i \neq j\}$ representing a set of finish-to-start precedence relations among activities. The duration of activity $i \in N$ is denoted by d_i, where $d_0 = d_{n+1} = 0$. There is a set of renewable resource $R = \{1, \ldots, m\}$ whose initial availability is the decision to be made before the project's start. The cost of providing one unit of resource k throughout the project is c_k. Each non-dummy activity $i \in N$ requires r_{ik} units of resource k throughout the project's execution phase. Note that $r_{0k} = r_{n+1,k} = 0, \forall\, k \in R$.

The resources are subject to breakdowns. We assume that there is a set $\Omega = \{1, \ldots, |\Omega|\}$ of distinct scenarios that support the uncertainty of the machine breakdowns. The number of units of resource k that breaks at time t under scenario ω is given by the integer parameter $f_{kt\omega}$ and the probability of scenario ω

is denoted by π_ω. We presume that $\sum_{\omega\in\Omega}\pi_\omega = 1$. In this setting, we assume that resources are hired from companies with information about their machines based on previous projects. The realization of scenarios is revealed when machines are available on the project's site, i.e., time zero, because machines are usually assigned by owner companies. Thus, scheduling the activities can be done based on the realized scenario.

In this problem, we assume that activities cannot be interrupted, i.e., preemption is not allowed. Therefore, in case of having insufficient resource units to carry on an activity, external resources must be employed. We assume there is a predetermined supplier for each resource based on a given contract to supply required resources at the beginning of the project. The cost of one unit of external resource k for a single time-unit is c'_k, which is supposed to be much higher than c_k/δ. It should be noted that external resources should not be employed when the project's resources are available.

The time horizon is discrete and denoted by $T = \{1, \ldots, \delta\}$ with δ being the project deadline, i.e., no activity can finish after this time moment. Let e_i be the earliest finish time, and l_i be the latest finish time of activity i. We introduce the following subsets of T:

$$T_i = \{e_i, \ldots, l_i\},$$

$$T_i^t = \{\tau \in T_i \,|\, t < \tau \leq t + d_i\},$$

$$T_\omega = \{\tau \in T \,|\, f_{k\tau\omega} > 0, \exists\, k \in R\}.$$

The goal is to determine the initial resource level and schedule the activities to minimize the total cost while respecting the resource and precedence constraints. We introduce binary variables $x_{it\omega}$, which is one if activity i finishes at time t under scenario ω and zero otherwise, and y_k which represents the initial availability of resource k. We also introduce auxiliary variable $z_{kt\omega}$, which represents shortfall of resource k at time t under scenario ω. In the following, we formulate the RACPSRA as a TSSPM in which variables y_k indicate the first stage or here-and-now variables and other variables represent the second-stage or wait-and-see variables.

$$\text{TSSPM} : \min \sum_{k\in R} c_k y_k + \sum_{\omega\in\Omega}\pi_\omega \sum_{k\in R}\sum_{t\in T} c'_k z_{kt\omega}, \tag{1}$$

s.t.

$$\sum_{t\in T_i} x_{it\omega} = 1, \quad \forall\, i \in N, \quad \forall\, \omega \in \Omega, \tag{2}$$

$$\sum_{t\in T_i} t x_{it\omega} \leq \sum_{t\in T_j} t x_{jt\omega} - d_j, \quad \forall\, (i,j) \in A, \quad \forall\, \omega \in \Omega, \tag{3}$$

$$\sum_{i\in N}\sum_{\tau\in T_i^t} r_{ik} x_{i\tau\omega} \leq y_k, \quad \forall\, k \in R, \quad \forall\, t \in T\backslash T_\omega, \quad \forall\, \omega \in \Omega, \tag{4}$$

$$\sum_{i\in N}\sum_{\tau\in T_i^t} r_{ik}x_{i\tau\omega} - (y_k - f_{kt\omega}) \le z_{kt\omega}, \quad \forall\, k \in R, \quad \forall\, \omega \in \Omega,$$

$$\forall\, t \in T_\omega, \tag{5}$$

$$x_{it\omega} \in \{0,1\}, \quad \forall\, i \in N, \quad \forall\, t \in T, \quad \forall\, \omega \in \Omega, \tag{6}$$

$$y_k \in \mathbb{Z}^+, \quad \forall\, k \in R, \tag{7}$$

$$z_{kt\omega} \in \mathbb{Z}^+, \quad \forall\, k \in R, \quad \forall\, t \in T, \quad \forall\, \omega \in \Omega. \tag{8}$$

The objective function (1), consisting of two terms, is to minimize the costs of the utilization of resources as well as the expected costs of hiring external resources. Constraints (2) are assignment constraints. Constraints (3) guarantee the precedence constraints among activities. Moreover, constraints (4) are related to the resources constraints for the periods at which resource k is available, whereas constraints (5) impose the resource constraints for periods at which resource k is under preventive maintenance operations. Finally, constraints (6)–(8) indicate the domain of variables where $\mathbb{Z}^+$ presents the set of non-negative integers.

4. A Decomposition-Based Algorithm

Möhring (1984) showed that the RACP is NP-hard, hence, the RACPSRA, which is a generalized version of the RACP, is NP-hard as well. As a result, the TSSPM developed in Sec. 3 is unable to solve optimally medium- and large-size instances of the RACPSRA using the CPLEX. Therefore, in this section, we develop a decomposition-based algorithm, which consists of two phases, to provide high-quality solutions in a reasonable runtime. In the first phase of this algorithm, we obtain all feasible combinations of the initial levels of resources. In the second phase, we solve the problem for each non-dominated combination of resource levels under all given scenarios to specify which resources need to be externally supplied and to what extent. The combination of resource levels resulting in the minimum expected value of resource costs is determined as the optimal solution. Table 1 summarizes the parameters used in this algorithm.

4.1. *The first phase*

In the first phase of our developed algorithm, we aim to provide lower and upper bounds to $y_k, \forall\, k \in R$. To this end, we initially adjust $\overline{y_k} = y_k^{\text{det}}$, where y_k^{det} is obtained by solving the integer linear programming model of the RACP model, shown as the deterministic model (DM) and developed by Möhring (1984). We then divide the main problem into $|\Omega|$ sub-problems, where sub-problem SP(ω) results in a schedule based upon the following model:

$$\text{SP}(\omega): \min \ \pi_\omega \sum_{k\in R}\sum_{t\in T} c'_k z_{kt\omega}, \tag{9}$$

Table 1. Description of parameters and variables of the DBA.

Parameter	Definition
$\overline{y_k}$	a numerical value of y_k
y_k^{det}	the optimal level of resource k obtained by solving the RACP
$y_k^{\min}$	the lower bound of resource k when other resources are unlimited
$y_l^{\min}(k)$	the lower bound of resource l in a schedule which aims to minimize only resource k
$y_k^{\max}$	the upper bound of resource k
NC	the total number of combinations of resource levels, considered in the second phase
y_k^q	the level of resource k in the qth combination of resource levels
CS_q	the qth combination of resource levels, $\text{CS}_q = (y_1^q, \ldots, y_m^q)$
$\text{Cost}(\text{CS}_q)$	cost of CS_q
CS	the set of all feasible combinations of resource levels
OCS	the set of all feasible combinations of resource levels in the non-descending order of total costs
$\text{SP}(\omega)$	the sub-problem relates to scenario ω
X_ω	the schedule obtained by solving $\text{SP}(\omega)$
X	the set of schedules obtained by solving all sub-problems
$\text{MP}(X, y_k^{\min})$	the TSSPM in which the set of schedules (X) and the level of resource k are known
SC^{MP}	the total cost of resource shortfall when $\text{MP}(X, y_k^{\min})$ is solved
CS_B	the combination of resource levels in the best-found solution

s.t.

$$\sum_{t \in T_i} x_{it\omega} = 1, \quad \forall\, i \in N, \tag{10}$$

$$\sum_{t \in T_i} t x_{it\omega} \leq \sum_{t \in T_j} t x_{jt\omega} - d_j, \quad \forall\, (i,j) \in A, \tag{11}$$

$$\sum_{i \in N} \sum_{\tau \in T_i^t} r_{ik} x_{i\tau\omega} \leq \overline{y}_k, \quad \forall\, k \in R, \quad \forall\, t \in T \backslash T_\omega, \tag{12}$$

$$\sum_{i \in N} \sum_{\tau \in T_i^t} r_{ik} x_{i\tau\omega} - (\overline{y}_k - f_{kt\omega}) \leq z_{kt\omega}, \quad \forall\, k \in R, \quad \forall\, t \in T_\omega, \tag{13}$$

$$x_{it\omega} \in \{0,1\}, \quad \forall\, i \in N, \quad \forall\, t \in T, \tag{14}$$

$$z_{kt\omega} \in \mathbb{Z}^+, \quad \forall\, k \in R, \quad \forall\, t \in T. \tag{15}$$

The goal of this model is to find an optimal schedule for scenario ω in which the level of each resource k is predefined as $\overline{y_k}$. The objective function (9) targets to minimize the costs of resource shortfall under scenario ω, and constraints (10) to (15) are very much alike to constraints of the developed TSSPM in Sec. 3.

Algorithm 1, called combinations set algorithm (CSA), indicates the general structure of the first phase of the DBA.

Algorithm 1. The CSA Algorithm.

Find values of y_k^{det} using the model DM
for $k = 1$ *to* m **do**
  Let $\overline{y_k} \leftarrow y_k^{\text{det}}$ and $\overline{y_l} \leftarrow \infty \, \forall \, l \neq k$
  Let *fsbl* $\leftarrow$ true
  while *fsbl* = true **do**
    for $\omega = 1$ *to* $|\Omega|$ **do**
      Solve SP(ω) and find X_ω
      if $X_\omega = \varnothing$ **then**
        Let *fsbl* $\leftarrow$ false
      end
    end
    if *fsbl* = true **then**
      Let $y_k^{\min} \leftarrow \overline{y_k}$, solve MP$(X, y_k^{\min})$ and find $y_l^{\min}(k), \forall \, l \in R \backslash k$
      Let $y_l^{\min}(k) \leftarrow (y_l^{\min}(k) + \lfloor \frac{\text{SC}^{\text{MP}}}{c_l} \rfloor), \forall \, l \in R \backslash k$
    end
    Let $\overline{y_k} \leftarrow (\overline{y_k} - 1)$
  end
end
if $m = 1$ **then**
  Let $y_k^{\max} \leftarrow (y_k^{\text{det}} + \max\{f_{kt\omega} | \forall \, \omega \in \Omega, \forall \, t \in T_\omega\})$
else
  Let $y_k^{\max} \leftarrow (\max\{y_k^{\min}(k') | \forall \, k' \in R, k' \neq k\}), \forall \, k \in R$
end
Let NC $\leftarrow \prod_{k=1}^{m} (y_k^{\max} - y_k^{\min} + 1)$
CS $\leftarrow \{\text{CS}_q | 1 \leq q \leq \text{NC}, \forall \, k, y_k^{\min} \leq y_k^q \leq y_k^{\max}\}$
for $q = 1$ *to* NC **do**
  Let Cost(CS$_q$) $\leftarrow \sum_{k=1}^{m} (y_k^q \times c_k)$
end
Generate OCS based on CS

One of the main tasks of the CSA is to find $y_k^{\min}$ and $y_k^{\max}$ for $\forall \, k \in R$. In the first step of CSA, the values of y_k^{det} $\forall \, k \in R$ are determined by means of solving the DM model. Next, a *for-loop* is applied to each resource $k \in R$ in which only resource k is assumed to be limited to $\overline{y_k} = y_k^{\text{det}}$ while others are unlimited. Then, a *while-loop* starts to find the minimum possible value of $\overline{y_k}$. In the first iteration of this *while-loop*, there is still at least one feasible solution to the problem under each scenario ($X_\omega \neq \varnothing$) because neither constraints (12) nor constraints (13) are violated. Therefore, the binary variable *fsbl* takes true. In each iteration of the *while-loop*, $\overline{y_k}$ is reduced by one unit until no feasible solution is found for a sub-problem SP(ω) ($\exists \, \omega : X_\omega = \varnothing$), which might happen only due to the violation of constraints (12). Whenever a new improved feasible $\overline{y_k}$ is found, the lower bound

$y_k^{\min}$ is updated and the TSSPM is solved to find the minimum levels of other resources, i.e., $y_l^{\min}(k) : \forall\, k \in R$.

It should be noted that $\mathrm{MP}(X, y_k^{\min})$ model is solved regarding a specific schedule. Hence, there might be other schedules with the same value of $y_k^{\min}$ in which a $y_l^{\min}(k)$ is higher, and lower costs of resource shortfall may lead to lower total costs. It would happen when the total cost of the shortfall is higher than the cost of providing one unit of resource l, i.e., when $\mathrm{SC}^{\mathrm{MP}} > c_l$. Consequently, we define $y_l^{\min} = y_l^{\min}(k) + \lfloor \frac{\mathrm{SC}^{\mathrm{MP}}}{c_l} \rfloor$ to have a valid lower bound to the level of resource $l : \forall\, l \in R \backslash k$.

Afterward, we continue by calculating the upper bounds of resource levels. In the case where we address only one type of constrained resource, the upper bound equals the initial level of the resource plus the maximum number of its failures in all scenarios. In other cases with more than one resource, $y_k^{\max}$ is calculated as follows:

$$y_k^{\max} = \max\{y_k^{\min}(l) \,|\, \forall\, l \in R, l \neq k\}. \tag{16}$$

Having the value of $y_k^{\min}$ and $y_k^{\max}$ for all resources, we can find all possible combinations of different resource levels. We denote the total number of combinations by NC, which is equal to $\prod_{k=1}^{m}(y_k^{\max} - y_k^{\min})$. We also indicate each combination and the level of resource k in that combination as CS_q and y_k^q, respectively, where $1 \leq q \leq \mathrm{NC}$. Therefore, the kth combination can be shown as $\mathrm{CS}_q = (y_1^q, \ldots, y_k^q, \ldots, y_m^q)$ in which $y_k^{\min} \leq y_k^q \leq y_k^{\max}$. After that, we calculate the cost of each combination ($\mathrm{Cost}(\mathrm{CS}_q)$) according to the cost of supplying the resources as $\sum_{k=1}^{m} c_k y_k^q$, and this step is repeated for all combinations. In the end, we sort the combinations in ascending order of their costs and generate the set OCS.

4.2. *The second phase*

In this phase of the DBA, we use set OCS to find the best feasible solution to the problem. Algorithm 2 shows the procedure of this phase, which is called the BFS algorithm. In this algorithm, we consider BOV as the objective value of the best-found solution, BRC as the cost of resource level corresponding to the best-found combination of resource levels, and BSC as the expected shortfall cost corresponding to the best-found solution.

In this phase, we have several combinations in which the resource levels are known upfront. For each element of the set OCS, we must solve the TSSPM. We begin with $\mathrm{BRC} = 0$ and $\mathrm{BOV} = \mathrm{BSC} = M$, where M implies a sufficiently large number. Next, counter q is set to 1, corresponding to the lowest cost case, and the following iterative procedure begins.

We first adjust $\overline{y_k} = y_k^q$, divide the model into $|\Omega|$ sub-problems and solve each. Next, we obtain BOV as the objective value of the best solution, consisting of two parts: BRC and BSC. The variable Sum shows the shortfall cost of each sub-problem, added to the cost of the corresponding combination ($\mathrm{Cost}(\mathrm{CS}_q)$) and compared to the objective value of the best-found solution (BOV). Each iteration

Algorithm 2. The BFS Algorithm.

Let $q \leftarrow 1$, BRC $\leftarrow 0$, BOV $\leftarrow M$ and BSC $\leftarrow M$ **while** $q \leq$ NC **do**
  for $k = 1$ *to* m **do**
    Let $\overline{y_k} \leftarrow y_k^q$
  end
  if Cost(CS$_q$) $\leq$ BOV **then**
    Let Sum $\leftarrow 0$ and *fsbl* $\leftarrow$ true **for** $\omega = 1$ *to* $|\Omega|$ **do**
      Solve SP(ω) **if** $X_\omega \neq \varnothing$ **then**
        Let Sum $\leftarrow$ (Sum + objective value(SP(ω))) **if** Sum + Cost(CS$_q$) > BOV **then**
          break (Dominated)
        end
      else
        fsbl $\leftarrow$ false break (Infeasible)
      end
    end
    if *fsbl* = true and Sum + Cost(CS$_q$) < BOV **then**
      Let BRC $\leftarrow$ Cost(CS$_q$) Let BSC $\leftarrow$ Sum Let BOV $\leftarrow$ (BRC + BSC)
    end
  else
    Stop
  end
  Let $q \leftarrow q + 1$
end

in which BOV is lower than the calculated cost value is called the "Dominated" case. Facing a "Dominated" case, we stop the corresponding iteration and go to the next one because that iteration would not improve the final objective value.

In an iteration where all the sub-problems are feasible, and no "Dominated" case is found, the obtained values for objective functions are considered as the best values and BRC, BSC and BOV are updated. Besides, if in an iteration, at least one "Infeasible" sub-problem is found, that iteration stops, and we go to the next one.

The algorithm stops when we achieve an iteration in which the cost of the resources combination is higher than the objective value of the best-found solution. Since the combinations are sorted in the non-descending order of their costs, in the next iteration, no improvement can be achieved. On account of this, the algorithm stops and reports the best-found solution.

Theorem. *The DBA finds the optimal solution of the RACPSRA for* $m \leq 2$.

The proof is provided in Appendix A.

4.3. *Illustration*

In this section, we implement the DBA on an instance to provide better clarification. The example project has ten activities, two resources and three scenarios, and the finish-to-start precedence relations among the activities are depicted in Fig. 1. Tables 2 and 3 summarize other parameters of activities and scenarios, respectively.

The costs of internal and external resources are provided in Table 4. Moreover, the deadline of the project (δ) is considered to be 28.

First, we solve this problem using the DM model and obtain the optimal levels of the resources as $y_1^{\text{det}} = 12, y_2^{\text{det}} = 17$. Then, we need to calculate the lower and upper bounds of resource levels. The steps that need to be taken for this phase are summarized in Table 5 in which false indicates there is no feasible solution. After that, the obtained lower and upper bounds are given in Table 6.

Having the lower and upper bounds, we then generate all the possible combinations of the resource levels. We have $7 \times 12 = 84$ combinations to calculate

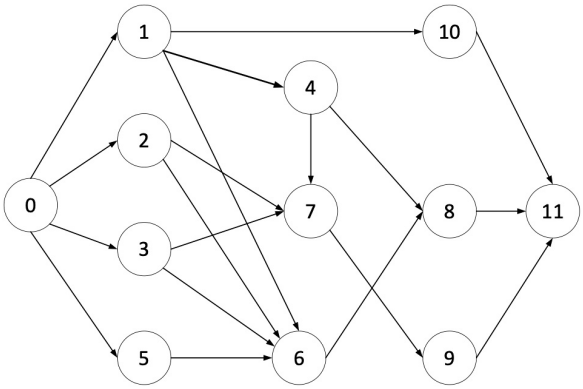

Fig. 1. Example project.

Table 2. Activity-related parameters of the example.

Activity (i)	d_i	r_{i1}	r_{i2}
1	1	1	8
2	3	1	9
3	8	3	9
4	1	9	2
5	6	2	3
6	4	10	8
7	7	6	7
8	6	3	8
9	9	2	5
10	4	1	2

Table 3. Scenario-related parameters of the example.

	ω_1			p_{ω_1}
T_{ω_1}	20	22	25	
k	1	2	2	0.4
$f_{k\tau\omega_1}$	5	2	4	
	ω_2			p_{ω_2}
T_{ω_2}	12	20	21	
k	1	1	2	0.3
$f_{k\tau\omega_2}$	4	4	8	
	ω_3			p_{ω_3}
T_{ω_3}	1	20	21	
k	2	1	1	0.3
$f_{1\tau\omega_3}$	3	6	2	

Table 4. Resource-related parameters of the example.

Resource 1		Resource 2	
c_1	c'_1	c_2	c'_2
28	4	33	5

Table 5. The steps taken to achieve the lower and upper bounds of resource levels.

Calculation of $y_1^{\min}$ and $y_2^{\min}(1)$							
Step	$\overline{y_1}$	SP(ω_1)	SP(ω_2)	SP(ω_3)	$\lfloor \frac{\mathrm{SC}^{\mathrm{MP}}}{c_2} \rfloor$	$y_2^{\min}(1)$	$y_1^{\min}$
1	12	0	0	0	$\lfloor \frac{4.519}{33} \rfloor$	29	12
2	11	9.58	5.99	8.43	$\lfloor \frac{28.52}{33} \rfloor$	26	11
3	10	11.18	7.19	9.64	$\lfloor \frac{29.83}{33} \rfloor$	26	10
4	9	false	—	—	—	26	10
Calculation of $y_2^{\min}$ and $y_1^{\min}(2)$							
Step	$\overline{y_2}$	SP(ω_1)	SP(ω_2)	SP(ω_3)	$\lfloor \frac{\mathrm{SC}^{\mathrm{MP}}}{c_1} \rfloor$	$y_1^{\min}(2)$	$y_2^{\min}$
1	17	0	0	0	$\lfloor \frac{2.41}{28} \rfloor$	16	17
2	16	0	0	0	$\lfloor \frac{2.41}{28} \rfloor$	16	16
3	15	0	0	0	$\lfloor \frac{2.41}{28} \rfloor$	16	15
4	14	false	—	—	—	16	15

their costs. After sorting the combinations in ascending order of their costs, the CSA algorithm is finished.

In the next phase, we use the output of the CSA algorithm as the input to the BFS algorithm and follow the steps shown in Table 7.

Table 6. The lower and upper bounds of the resource levels.

$y_1^{\max}$	$y_1^{\min}$	$y_2^{\max}$	$y_2^{\min}$
16	10	26	15

Table 7. The steps of BFS algorithm for the example project.

q	CS_q		$Cost(CS_q)$	$SP(\omega_1)$	$SP(\omega_2)$	$SP(\omega_3)$	BRC	BSC	BOV
	y_1^q	y_2^q							
1	10	15	775		Infeasible		0	10,000	10,000
2	11	15	803		Infeasible		0	10,000	10,000
3	10	16	808		Infeasible		0	10,000	10,000
4	12	15	831		Infeasible		0	10,000	10,000
5	11	16	836		Infeasible		0	10,000	10,000
6	10	17	841		Infeasible		0	10,000	10,000
7	13	15	859		Infeasible		0	10,000	10,000
8	12	16	864		Infeasible		0	10,000	10,000
9	11	17	869		Infeasible		0	10,000	10,000
10	**10**	**18**	874	11.8	9.582	9.640	874	30.40	904.40
11	14	15	887		Infeasible		874	30.40	904.40
12	13	16	892		Infeasible		874	30.40	904.40
13	12	17	897	7.984	Dominated		874	30.40	904.40
14	11	18	902	9.581	Dominated		874	30.40	904.40
15	10	19	907		Stop		874	30.40	**904.40**

As shown in Table 7, at the beginning of the BFS algorithm, we assign a big number to BOV and BSC and let BRC = 0. Then, in all steps 1 to 9, the first sub-problem ($SP(\omega_1)$) reaches infeasibility, but in step 10, all the sub-problems are solved, resulting in BOV = 904.40. We then skip steps 11 and 12 due to the infeasibility of sub-problems. In steps 13 and 14, the sum of the objective value of the first sub-problem and the cost of the corresponding combination is higher than the available best-found solution. Therefore, the "Dominated" case takes place, and we stop solving other sub-problems in that iteration. Eventually, in step 15, the

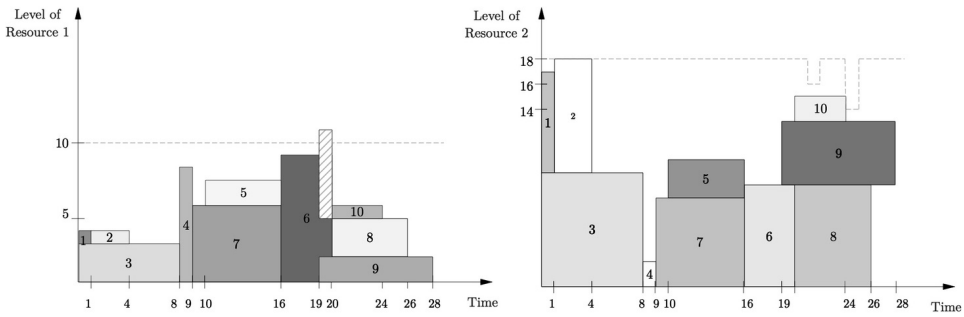

Fig. 2. The obtained schedule of scenario 1.

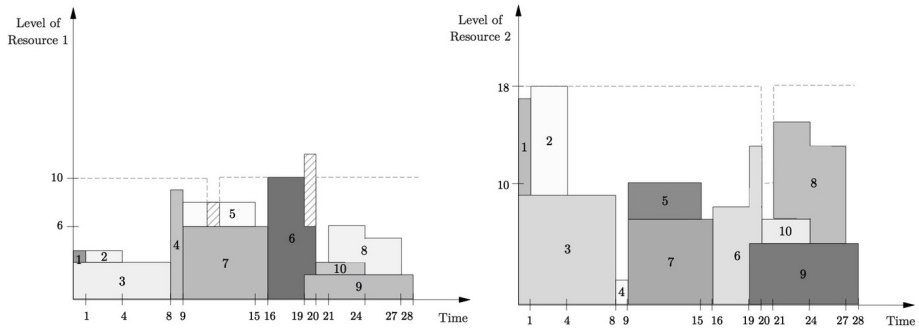

Fig. 3. The obtained schedule of scenario 2.

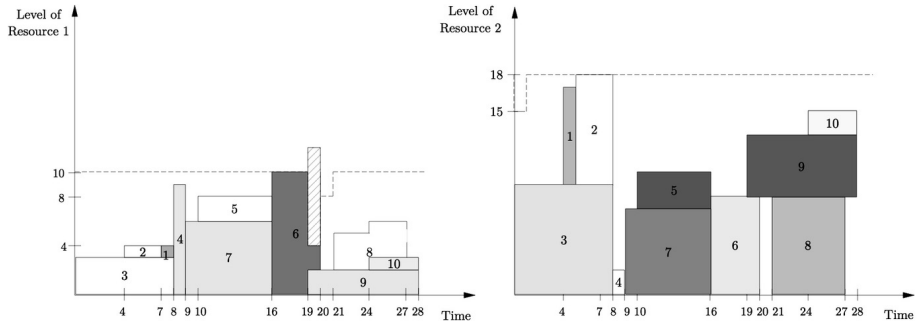

Fig. 4. The obtained schedule of scenario 3.

cost of the corresponding combination is higher than the best-found solution; the algorithm stops, and the final solution is achieved as $y_1 = 10$ and $y_2 = 18$.

Figures 2 to 4 represent the Gantt chart corresponding to the optimal schedule of each scenario, where the shortfalls of resources are shown by a crosshatch pattern.

5. Scenario Generation and Reduction Methods

In this section, first, we describe how the scenario set Ω is generated for the RACP-SRA, called the scenario generation method. Next, we explain how this set can be diminished and mapped to a smaller appropriate set of scenarios, named the scenario reduction method.

5.1. *Scenario generation method*

Each renewable resource has a nonzero probability of breakdown at a period, which we assume that such breakdowns can be estimated based on the life cycle and the performance of machines (Van den Berg and Aardal, 2015). Without loss of generality, we also assume that this probability is time-independent. On account of this, the breakdown probability of each unit of a resource follows a Bernoulli

distribution, and the whole resource has a binomial distribution (Jensen and Sun, 2013). Each scenario indicates a set of periods in which some resources fails to service casually. To generate the set of scenarios for an instance of the RACPSRA, we use the parameters described in Table 8.

In our scenario generation method, each scenario is defined by an ordered pair (t_θ, k_θ), where $t_\theta \in T_\omega$ denotes the period in which resource $k_\theta \in R_\omega$ has $f_{k_\theta t_\theta \omega}$ units reduction in its level. $R_\omega = \{k_1, k_2, \ldots, k_s\}$ is the set of resources which corresponds to T_ω that an arbitrary amount of reduction happen in their levels. It should be noted that there might be some repetitive elements in this set. We also assume that the total number of the ordered pairs in all scenarios is equal to s.

To define the probability of each scenario, we first need to estimate the probability of a f-units reduction in resource k. As mentioned in Sec. 3, in the TSSPM, we defined the resource levels as decision variables, but here we assume that the level of resource k is approximately equal to y_k^{det}. The values of y_k^{det} are determined using Eq. (17) and values of y_t^k, obtained by solving the DM model:

$$y_k^{\text{det}} = \max_{t \in T} \{y_t^k\}; \quad \forall\, k \in R. \tag{17}$$

Considering the above assumption, we can calculate the probability of f units reduction in the level of resource $k \in R_\omega$ at time t in scenario $\omega \in \Omega$ as follows:

$$P(f_{kt\omega} = f) = \binom{y_k^{\text{det}}}{f} \times (p_k)^f \times (1 - p_k)^{y_k^{\text{det}} - f}; \quad \forall\, t \in T_\omega, \quad \forall\, k \in R_\omega, \quad \forall\, \omega \in \Omega. \tag{18}$$

Now, we assume that the maximum reduction in the level of resource k is $f_k^{\max}$. We also presume that the probability of having no reduction in the resource levels is zero ($P(f_{kt\omega} = 0) = 0$). To calculate the probability of resource reduction, we use the conditional binomial distribution as follows:

$$P(f_{kt\omega} = f \,|\, 1 \le f_{kt\omega} \le f_k^{\max}) = \frac{\binom{y_k^{\text{det}}}{f} \times (p_k)^f \times (1 - p_k)^{y_k^{\text{det}} - f}}{\sum_{\lambda=1}^{f_k^{\max}} \binom{y_k^{\text{det}}}{\lambda} \times (p_k)^\lambda \times (1 - p_k)^{y_k^{\text{det}} - \lambda}}. \tag{19}$$

Following the given formulas, the probability of scenario $\omega \in \Omega$ is calculated by the following equation:

$$\pi_\omega = \prod_{\theta=1}^{s} P(f_{k_\theta t_\theta \omega}). \tag{20}$$

Table 8. Description of parameters used in scenario generation method.

Parameter	Definition
s	total number of ordered pairs defined as (t_θ, k_θ) in scenario ω
y_t^k	the total usage of active activities from resource k at time t
p_k	the probability of one unit shortfall of resource k at any time unit
$f_k^{\max}$	the maximum reduction in the level of resource k
R_ω	the set of resources which encounter shortfall in scenario ω

5.2. *Scenario reduction method*

In this section, we present the scenario reduction method to decrease the number of possible scenarios and the complexity of the problem. The parameters used in this method are described in Table 9.

This algorithm works based on the Kantorovich distance concept. This measure is also used by Bruninx and Delarue (2016) and determines the optimal reduced set of scenarios by computing the probabilistic distance between two scenario sets. We also need to use the Monge–Kantorovich mass transportation problem (Heitsch and Römisch, 2003). This problem is proposed as the well-known linear transportation problem, described below, for two discrete sets of Ω and Ω'. These two sets have finite probability distributions $Q = \sum_{\omega=1}^{|\Omega|} \pi_\omega$ and $Q' = \sum_{\omega'=1}^{|\Omega'|} \pi_{\omega'}$, respectively,

$$\min \sum_{\omega,\omega'} c(\omega,\omega')\eta_{\omega\omega'}, \tag{21}$$

s.t.

$$\sum_{\omega} \eta_{\omega\omega'} = \phi_{\omega'}, \quad \forall\, \omega' \in \Omega', \tag{22}$$

$$\sum_{\omega'} \eta_{\omega\omega'} = \pi_\omega, \quad \forall\, \omega \in \Omega, \tag{23}$$

$$\eta_{\omega\omega'} \geq 0, \quad \forall\, \omega' \in \Omega'. \tag{24}$$

In this model, we aim at eliminating some scenarios from set Ω and transferring their probability space to the remaining ones while minimizing the cost of transferring. The cost is denoted by $c(\omega,\omega')$, which accounts for the distance between two scenarios. $\eta_{\omega\omega'}$ represents the amount of probability space that is transferred to scenario ω' in case of omitting scenario ω. We assume that J is the set of scenarios that is remained from set Ω such that $J = \Omega\backslash\Omega'$. In the case where the space probability of a scenario is completely transferred to another scenario, we can rewrite

Table 9. Description of parameters used in scenario reduction method.

Parameter	Definition
$\Omega' = 1,\ldots,\lvert\Omega'\rvert$	the set of reduced scenarios with index ω'
Q	the finite probability space of set Ω
Q'	the finite probability space of set Ω'
ω_0	a fixed element of set Ω
AS_t^k	the set of activities that use resource k at time period t
NIA_t^k	number of activities that the reduction in resource k at time t changes their start time
$\mathrm{suc}(i)$	the set of direct and indirect successors of activity i
π_ω	the probability of scenario ω in probability space Q
$\phi_{\omega'}$	the probability of scenario ω' in probability space Q'

the Kontorovich distance measure as follows:

$$D = \min\left\{\sum_{\omega\in J} \pi_\omega \cdot \min_{\omega'\in\Omega'} c(\omega,\omega')\right\}. \tag{25}$$

Equation (25) represents a version of the set covering problem in which the set of scenario Ω is covered by two sets J and Ω' with the coverage cost D. Since the set covering problem is NP-hard (Chvatal, 1979), heuristic algorithms must be developed to solve it in a reasonable time. In this regard, Dupacová *et al.* (2003) introduced the fast forward and backward algorithms. The fast forward algorithm is faster in case of having a small-size set of reduced scenarios (Heitsch and Römisch, 2003). Therefore, we apply this method to our problem. At the outset of this algorithm, we start with an empty set and choose a specific number of scenarios one by one. Dupacová *et al.* (2003) presented the objective function of this model as given in Eq. (31), where ω_0 is a fixed element in set Ω, and h is a positive continuous non-decreasing function where $h(0) = 0$,

$$c(\omega,\omega') = \max\{1, h(\|\omega-\omega_0\|), h(\|\omega'-\omega_0\|)\}\cdot\|\omega-\omega_0\|. \tag{26}$$

We also define the distance between two scenarios ω and ω' by a function in which G_ω is the score of scenario ω,

$$c(\omega,\omega') = |G_\omega - G_{\omega'}|. \tag{27}$$

Furthermore, we define the distance between two scenarios by a function in which G_ω is the score of scenario ω. In order to calculate the scenario scores, we need to investigate how reducing the level of resource k in scenario ω affects the model. This reduction impacts on two main factors: (a) the total cost of the model, and (b) the scheduling of the activities. Considering these factors, we can formulate the score of scenario ω as follows:

$$G_\omega = \sum_{\theta=1}^{s} \frac{f_{k_\theta t_\theta \omega}}{y_{k_\theta}^{\det} - y_{t_\theta}^{k_\theta} + 1} \times \frac{c'_{k_\theta}}{c_{k\theta}} \times \frac{\mathrm{NIA}_{t_\theta}^{k_\theta}}{n}, \quad \forall\,\omega\in\Omega, \tag{28}$$

where $\frac{f_{k_\theta t_\theta \omega}}{y_{k_\theta}^{\det} - y_{t_\theta}^{k_\theta} + 1} \times \frac{c'_{k_\theta}}{c_{k\theta}}$ denotes the effect of the resource level reduction on the total cost. We compare the reduction $f_{k_\theta t_\theta \omega}$ to the $y_{k_\theta}^{\det} - y_{t_\theta}^{k_\theta} + 1$ to measure the cost impact. The lower consumption of resource k_θ at time t_θ, the more freedom for reducing the resource level at that time and the lower shortfall cost. We also add one unit to the denominator to prevent it from getting zero. Besides, $\frac{\mathrm{NIA}_{t_\theta}^{k_\theta}}{n}$ captures the effect of resource reduction on scheduling. To obtain $\mathrm{NIA}_{t_\theta}^{k_\theta}$, we determine the activities we have difficulty in performing them at time t_θ in case of reducing the level of resource k_θ. Thereafter, we specify their successors. If there is more than one activity using resource k_θ at the same time, we show them with set $\mathrm{AS}_{t_\theta}^{k_\theta}$, and then divide the total number of successors by the number of activities in the mentioned

set:

$$\mathrm{NIA}_{t_\theta}^{k_\theta} = \frac{\sum_{\forall\, i \in \mathrm{AS}_{t_\theta}^{k_\theta}} |\mathrm{suc}(i)|}{|\mathrm{AS}_{t_\theta}^{k_\theta}|}. \tag{29}$$

Having the score of each scenario, we then proceed with the scenario reduction algorithm as it is provided by Heitsch and Römisch (2003).

6. Computational Results

To investigate the performance of the developed algorithms, we generated 180 random test instances. The algorithms were coded in Visual C++ and run on a portable computer with an Intel Core i7 (3.40 GHz, 32 GB of RAM) under Windows 10 operating system. We also used the IBM ILOG CPLEX 12.8 to run our models.

6.1. *Experimental setup*

We generated the random instances by RanGen software (see Vanhoucke *et al.* (2003)) based upon three main parameters: (1) number of activities, (2) number of resources, and (3) network complexity. The network complexity of a test instance implies the number of direct and indirect precedence relations among activities divided by the maximum number of possible precedence relations in that instance. The number of activities and resources are taken from the sets $\{20, 25, 30\}$ and $\{1, 2\}$, respectively. The network complexity is considered as OS $\in \{0.4, 0.6, 0.8\}$. Finally, for each combination of the aforementioned parameters, we generated ten random instances resulting in 180 test instances in total.

As the first step, we solved all the instances using the deterministic RACP model (developed by Demeulemeester (1995)) by the CPLEX and obtained the y_k^{det} values. We found that considering all periods in which resource failure might happen significantly enlarges the solution space and makes the problem intractable. Therefore, we only confined failure periods to a particular set, called the set of critical periods (T_{cr}). To create this set, we considered periods in which $\exists\, k; y_t^k \in [y_t^{\mathrm{det}} - 2, y_t^{\mathrm{det}}]$ and one resource might fail to serve in them. Consequently, we considered an ordered pair (t_θ, k_θ) for each period and its corresponding resource, where set T_{cr} contains all the ordered pairs. We also assumed an upper limit for the reduction of each resource level in such a way that $f_{k_\theta t_\theta \omega} \leq \frac{y_{k_\theta}^{\mathrm{det}}}{2}$.

To generate each scenario, we randomly selected eight ordered pairs from set T_{cr}, and for each chosen pair we took a random value for $f_{kt\omega}$ from discrete uniform distribution $[1, \frac{y_{k_\theta}^{\mathrm{det}}}{2}]$. Then, to obtain the probability of each scenario, we first took a random value from $U(0.7, 0.8)$ and then calculate π_ω and G_ω as explained in Sec. 5.

Finally, for each test instance, 50 scenarios are generated, and their probabilities were normalized to achieve the complete probability space. We also applied the scenario reduction algorithm and reduced the number of scenarios to 50% (25 scenarios) and 25% (12 scenarios) of the initial amount.

6.2. *Computational performance*

In this section, we provide the computational results of the TSSPM and the DBA. For this purpose, we set the time limit to 3,600 seconds in the CPLEX, and the stopping criterion is reaching a relative *gap* of at least 0.0001 between the best-found integer solution and the best lower bound of the remaining nodes. This *gap* is denoted by the percentage of the relative gap, shown as PRG, in the following.

In the case of solving the instances with the all scenarios, the complexity of the problem is considerably high. Thus, the CPLEX stops at the given time limit before reaching the optimal solution. In this case, we used the average of PRG (APRG) to compare the results of different instances. Table 10 represents APRG on each group of instances, which are categorized based on the number of activities (n), the number of precedence relations (OS), as well as the number of resources (m) with 50 scenarios.

As can be seen, increasing the number of activities and the number of resources leads to a higher value of APRG, which is caused by the higher complexity of the problem. In terms of the number of precedence relations, we expected that by increasing this value, the slack of activities increases, and the problem gets easier to solve. Despite our expectations, however, we found that in the time limit of one hour, APRG gets larger when OS increases. The reason might well be due to the fact that the CPLEX is not able to find a better (larger) lower bound for the instances with larger OS, hence, the APRG increases.

Tables 11 and 12 summarize the APRG when the number of scenarios is diminished to 25 and 12, respectively.

According to the above tables, the same trend takes place when the number of scenarios is reduced.

We also categorized the instances based on the number of scenarios and activities. As a result, we have 180 instances equally categorized in three groups with 20, 25, and 30 activities, in which three subgroups, with 12, 25, and 50 scenarios, are involved. We show the results of these groups in Fig. 5. This figure signifies that decreasing the number of scenarios leads to a reduction in APRG. Since reducing the number of scenarios decreases the number of constraints and decision variables,

Table 10. The APRG of TSSPM for the complete version of scenarios.

m	OS	n		
		20	25	30
1	0.4	8.7	12.9	13.4
	0.6	10.5	16.2	21.2
	0.8	17.0	26.0	33.6
2	0.4	19.3	19.6	20.0
	0.6	23.6	24.2	28.2
	0.8	28.8	36.5	37.9

Table 11. The APRG of TSSPM for 25 scenarios.

m	OS	n		
		20	25	30
1	0.4	4.7	10.4	11.3
	0.6	5.1	12.1	18.1
	0.8	9.2	21.9	30.7
2	0.4	15.7	16.9	17.0
	0.6	16.0	19.9	25.5
	0.8	21.3	29.1	33.9

Table 12. The APRG of TSSPM for 12 scenarios.

m	OS	n		
		20	25	30
1	0.4	1.5	5.6	6.1
	0.6	1.5	8.8	13.0
	0.8	2.2	12.0	20.5
2	0.4	11.3	13.1	13.3
	0.6	11.8	15.6	21.5
	0.8	13.2	21.7	28.3

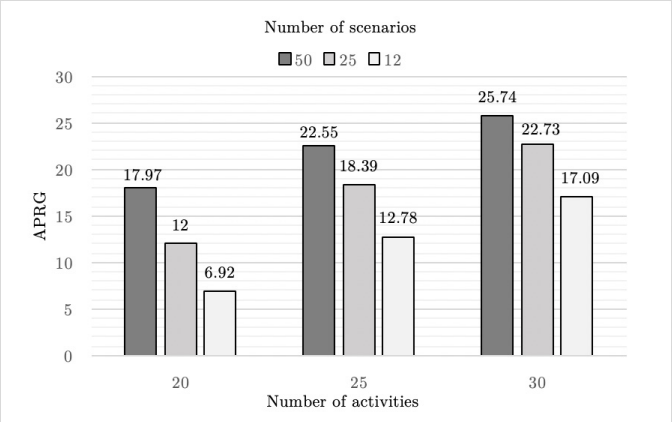

Fig. 5. The comparison of APRG for instances with reduced number of scenarios.

the problem complexity is also reduced, and in a given time limit, a solution with a lower PRG can be achieved.

It is worth mentioning that when the TSSPM is employed, only 3% of test instances, including those address the reduced version set of scenarios, could reach zero PRG in less than 3,600 seconds. Table 13 shows the average runtime of solving the instances using the TSSPM. In each group, the number of instances that reached optimality within the given time limit is reported inside the parenthesis.

Table 13. The average runtime of the TSSPM.

$\|\Omega\|$	n		
	20	25	30
50	3524 (2)	3550 (1)	3600 (0)
25	3256 (7)	3548 (1)	3600 (0)
12	3057 (10)	3473 (2)	3600 (0)

We implemented the DBA on the 180 instances using the complete version of scenarios and provided the average runtime (in seconds) in Table 14.

The DBA is capable of solving all the instances in less than an hour. The group of instances with $n = 30$, $m = 2$, and OS $= 0.4$ had the highest runtimes with an average of 2,300 seconds. This group accounts for 5% of all the instances. Half of the all instances took less than 100 seconds to obtain the optimal solution, and 40% of them are solved between 100 to 1,000 seconds. The rest of the instances (5% of the total) had a runtime of almost 1,800 seconds, which is still half of the time spent solving the TSSPM.

According to Table 14, it can also be concluded that increasing the number of activities and the number of resources causes an increase in the runtime of the DBA. However, increasing the precedence relations has diminished this measure. The reason is that the larger number of relations decreases the slack of activities; their possible start and finish times are declined.

We also implemented the DBA in conjunction with the scenario reduction method, where the number of scenarios was 25 and 12. These results are presented in Tables 15 and 16.

As can be seen in Table 15, in case of a 50% reduction in the number of scenarios, 55% of the instances were solved in less than 100 seconds, while the rest reached the optimal solution between 100 to 1,000 seconds.

According to the data given in Table 16, in case of decreasing the scenarios to 25% of their initial number, 67% of the instances found the optimal solution in less

Table 14. The APRG of TSSPM for complete scenarios.

m	OS	n		
		20	25	30
1	0.4	59.4	225.3	702.8
	0.6	25.2	92.3	567.7
	0.8	16.1	45.4	98.3
2	0.4	179.1	785.8	2289.0
	0.6	83.0	641.8	1892.4
	0.8	54.1	79.0	238.5

Table 15. The APRG of TSSPM for 25 scenarios.

m	OS	n		
		20	25	30
1	0.4	30.2	104.7	318.3
	0.6	12.6	46.6	295.2
	0.8	7.6	21.7	50.5
2	0.4	92.3	375.5	996.5
	0.6	42.3	329.9	834.5
	0.8	30.1	39.6	118.2

Table 16. The APRG of TSSPM for 12 scenarios.

m	OS	n		
		20	25	30
1	0.4	15.8	53.7	169.6
	0.6	6.5	24.0	154.8
	0.8	3.7	11.7	26.3
2	0.4	49.4	224.5	618.3
	0.6	21.2	174.5	493.6
	0.8	14.9	19.8	64.9

than 100 seconds. The remaining instances were also solved optimally between 100 to 1,000 seconds.

Figure 6 depicts the runtime of the DBA in different groups of instances. As can be seen, the number of scenarios has a significant effect on the algorithm's runtime. A 50% reduction in the number of scenarios has resulted in a 50% decrease in the runtime.

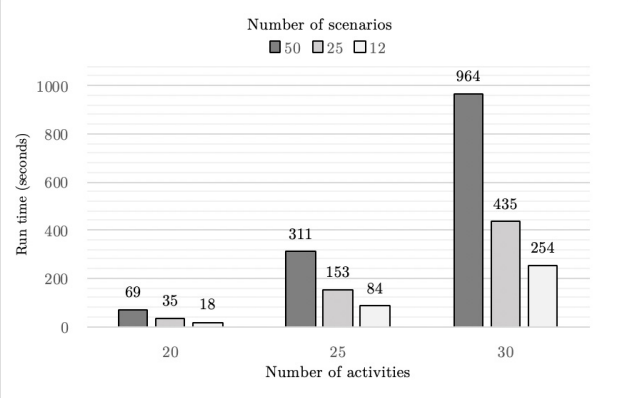

Fig. 6. The comparison of runtime of TSSPM and the DBA.

In order to show the value of stochastic solutions (VSS), we consider each instance with $n = 20$ activities and $|\omega| = 12$ and 25 for which we have their optimal solutions. To obtain the VSS of each instance, we first consider the most probable scenario as the only possible scenario of the instance and solve it using the TSSPM to find resource levels (we ignore the obtained schedule). Given the values of resource levels, we rerun the TSSPM with all scenarios to find corresponding schedules and the objective function value. Next, we compare it with the optimal objective value, obtained by the TSSPM and considering all scenarios. For each instance, we divide the difference between the two aforementioned objective values, which is called VSS, by the optimal objective value of the instance multiplied by 100. The average values of this measure, shown as the average percent of VSS (APVSS), are shown in Table 17. As can be seen, the larger values of m and $|\Omega|$ are considered, the more APVSS is achieved.

6.3. *Comparative results*

In this section, we compare the performance of the DBA and the TSSPM using the APD, indicating the average percentage deviation of the results obtained using the DBA than the TSSPM. The APD is calculated as follows:

$$\text{APD} = \frac{\sum_{i=1}^{60} \frac{\text{DBA}_i - \text{TSSP}_i}{\text{TSSP}_i} \times 100}{60}. \tag{30}$$

The results are summarized in Table 18.

The negative values in Table 18 indicate the better performance achieved by the DBA compared to the TSSPM. According to this table, the more the number of scenarios, the more superiority of the DBA. This signifies the outperformance of the DBA in more complex problems. Table 19 indicates for how many instances the

Table 17. The APVSS values.

$\|\Omega\|$	m	
	1	2
12	20.1	15.7
25	27.6	21.3

Table 18. The APD of results in one-hour time limit.

$\|\Omega\|$	n		
	20	25	30
50	−9.6%	−12.7%	−16.3%
25	−5.5%	−8.9%	−12.2%
12	−2.7%	−5.4%	−7.9%

Table 19. The number of instances that achieved better solutions by the DBA.

$\|\Omega\|$	n		
	20	25	30
50	54	60	60
25	49	56	60
12	40	50	60

Table 20. The number of instances with runtime of less than an hour.

$\|\Omega\|$	n		
	20	25	30
50	60	60	52
25	60	60	59
12	60	60	60

DBA could find a better solution than the TSSPM. It is worth mentioning that for no instance the DBA found a worse solution than the TSSPM.

Furthermore, in the version of complete scenarios, 97% of the instances achieved better solutions when they were solved by the DBA. However, this measure declines to 92% and 83% for the cases with 25 and 12 scenarios, respectively. The reason for this downward trend is the improvement that the scenario reduction method brings to the TSSPM. Generally, we can conclude that in the instances with a larger number of activities, the DBA performs better in all versions of the number of scenarios. This conclusion is more important when we observe that for almost all the instances, the runtime of the TSSPM was 3,600 seconds, but it could not obtain the desired convergence in this time limit. On the other hand, the runtime of the DBA is much less than the TSSPM, and only 1% of the instances took more than 3,600 seconds to achieve the stopping criterion. Table 20 shows the number of instances that could be solved within the one-hour time limit by the DBA. This table confirms that more than 99% of the all instanced have been solved optimally within one hour. Moreover, Table 21 indicates the percentage of instances that have been solved within each runtime interval.

Table 21. Time intervals of runtime in the DBA.

$\|\Omega\|$	Time intervals (seconds)		
	[0, 100]	[100, 1,000]	[1,000, 3,600]
50	50%	38%	12%
25	55%	45%	0%
12	67%	33%	0%
Total	57%	39%	4%

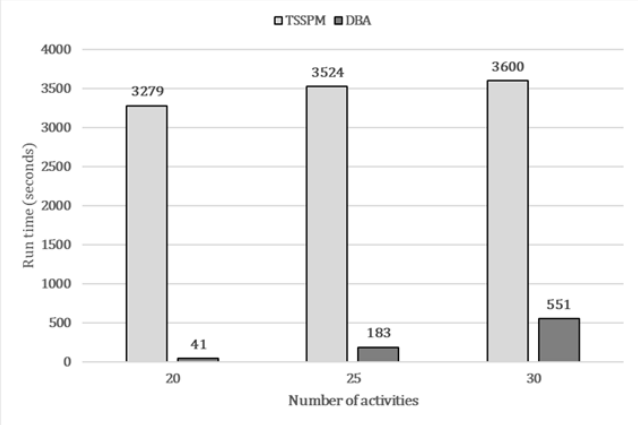

Fig. 7. The comparison of CPU runtimes based on the number of activities.

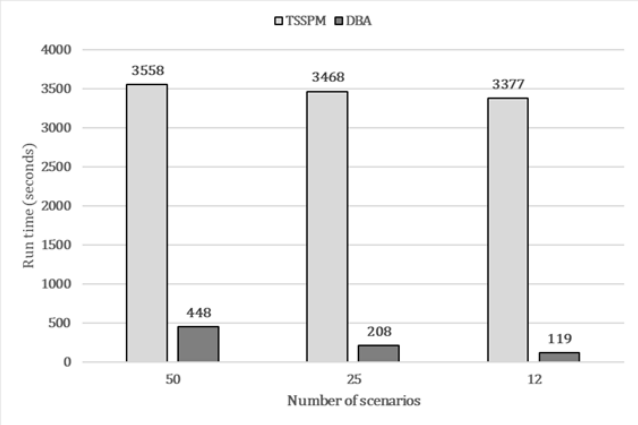

Fig. 8. The comparison of CPU runtimes based on the number of scenarios.

According to Table 21, the DBA creates a runtime less than 100 seconds for 57% of the instances, and only it takes more than one hour for 4% of them. Figures 7 and 8 compare the runtime of the two developed solution approaches based on the number of activities as well as the number of scenarios. Both figures show the superior performance of the DBA algorithm. The average runtime for the TSSPM is 3,467 seconds while this value is 775 seconds for the DBA. In other words, the average runtime was declined more than 77% in by the DBA.

7. Summary and Outlook on Future Research

In this paper, we studied the resource availability cost problem with stochastic resource availability. We defined a set of scenarios to deal with the uncertainty of

resource levels. To solve the problem, we formulated it as a two-stage stochastic programming model and used the CPLEX to solve it. We then found out that the model is unable to be solved in less than one hour, even for the small-size instances. Therefore, we developed a decomposition-based algorithm that is capable of solving the instances in a reasonable time. Furthermore, we proved our developed algorithms find optimal solutions when the number of resources of the problem is not more than two resources. The quality of the solutions obtained by the decomposition-based algorithm is better than the TSSPM in 91% of the instances.

For future studies, several directions can be considered. First, developing exact solution methods that can find the optimal solution in the general form ($m > 2$) is an interesting and challenging research topic. In addition to this, efficient heuristic or metaheuristic algorithms can also be developed to solve the problem. Besides, inflation can be added to the problem so that the cost of resources changes over time. Another assumption that can be interesting is the activity preemption. In other words, in the case of a shortfall in a resources, we have two options to address: supplying them externally or halting the activity and re-start it whenever the required resources are available.

Appendix A

We consider the problem into two cases and investigate them separately.

Case A: $m = 1$.

In this case, we know that y_1^{det} is the optimal schedule for the RACP. If we consider the resultant schedule of the RACP for the RACPSRA, we encounter shortfalls due to the abatement at the resource level. Moreover, if we adjust ($y_1^{\max}$) to $y_1^{\text{det}} + \max\{f_{1t\omega} \,|\, \forall\, \omega \in \Omega, \forall\, t \in T_\omega$, no shortfall takes place in the optimal solution. Consequently, increasing the resource level to a magnitude more than this would result in no cost reduction. As a result, the optimal solution to the problem would be less than $y_1^{\max}$, hence, the optimal value of y_1 would undoubtedly be in interval $[y_1^{\min}, y_1^{\max}]$ and the DBA is able to achieve it.

Case B: $m = 2$.

In this case, we must find lower and upper bounds for each resource. To this end, we first assume that the second resource is unlimited and minimize the first resource level based on the CSA algorithm, leading to a valid lower bound for the first resource ($y_1^{\min}$). This results in a set of schedules, called X^1, and each of them corresponds to a scenario. We are able to find an upper bound for the second resource level based on $y_1^{\min}$ and X^1, shown by $y_2^{\min}(1)$ and obtained through $\text{MP}(X^1, y_1^{\min})$. It must be noted that $y_2^{\min}(1)$ might not be a valid upper bound for y_2 because there likely exists another set of schedules (X^2) and resource level for y_2 which may give rise to a smaller total cost. This might happen when a higher level is addressed for y_2 resulting in a lower shortfall cost. As the worst case, suppose all shortfall costs relate to the second resource, and we aim at decreasing the shortfall

costs by means of increasing the second resource level. This trade-off leads to a better solution until $SC^{MP} \geq c_2$. Consequently, we establish a valid upper bound for y_2 as $y_2^{max} = y_2^{min}(1) + [\frac{SC^{MP}}{c_2}]$. Likewise, we can acquire y_2^{min} and y_1^{max} using the aforementioned process and exchanging the role of the two resources. The optimal values of y_1 and y_2 are within intervals $[y_1^{min}, y_1^{max}]$ and $[y_2^{min}, y_2^{max}]$, respectively. All possible combinations of y_1 and y_2 are considered in set CS, transformed into ordered set OCS based on ascending total costs. Therefore, the final solution of the BFS algorithm is the best solution among combinations of CS, and it would be the optimal solution.

References

Afruzi, EN, A Aghaie and AA Najafi (2018). Robust optimization for the resource constrained multi-project scheduling problem with uncertain activity durations. *Scientia Iranica*, 27(1), 361–376.

Afshar-Nadjafi, B (2014). Multi-mode resource availability cost problem with recruitment and release dates for resources. *Applied Mathematical Modeling*, 38, 5347–5355.

Bei, X, N Chatwattanasiri, DW Coit and X Zhu (2017). Combined redundancy allocation and maintenance planning using a two-stage stochastic programming model for multiple component systems. *IEEE Transactions on Reliability*, 66(3), 950–962.

Bruni, ME, LDP Pugliese, P Beraldi and F Guerriero (2017). An adjustable robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations. *Omega*, 71, 66–84.

Bruni, ME, LDP Pugliese, P Beraldi and F Guerriero (2018). A two-stage stochastic programming model for the resource constrained project scheduling problem under uncertainty. In *Proc. 7th Int. Conf. Operations Research and Enterprise Systems*, pp. 194–200. SciTePress.

Bruninx, K and E Delarue (Eds.) (2016). Scenario reduction techniques and solution stability for stochastic unit commitment problems. In *2016 IEEE Int. Energy Conf. (ENERGYCON)*, pp. 1–7. IEEE.

Chakrabortty, RK, A Abbasi and MJ Ryan (2019). A risk assessment framework for scheduling projects with resource and duration uncertainties. *IEEE Transactions on Engineering Management*, 69, 1917–1931.

Chvatal, V (1979). A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3), 233–235.

Coughlan, E, M Lübbecke and J Schulz (2015). A branch-price-and-cut algorithm for multi-mode resource leveling. *European Journal of Operational Research*, 245, 70–80.

Demeulemeester, E (1995). Minimizing resource availability costs in time-limited project networks. *Management Science*, 41, 1590–1598.

Dupacová, J, N Gröwe-Kuska and W Römisch (2003). Scenario reduction in stochastic programming. *Mathematical Programming*, 95, 493–511.

Fu, N, HC Lau and P Varakantham (2015). Robust execution strategies for project scheduling with unreliable resources and stochastic durations. *Journal of Scheduling*, 18(6), 607–622.

Habibi, F, F Barzinpour and S Sadjadi (2018). Resource-constrained project scheduling problem: Review of past and recent developments. *Journal of Project Management*, 3, 55–88.

Hartmann, S and D Briskorn (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207, 1–14.

Heitsch, H and W Römisch (2003). Scenario reduction algorithms in stochastic programming. *Computational Optimization and Applications*, 24(2–3), 187–206.

Herroelen, W and R Leus (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165(3), 289–306.

Hsu, C and D Kim (2005). A new heuristic for the multi-mode resource investment problem. *Journal of the Operational Research Society*, 56, 406–413.

Jardine, A and A Tsang (2013). *Maintenance, Replacement, and Reliability: Theory and Applications*, 2nd edn. CRC Press.

Jensen, TTM and Q Sun (2013). Absenteeism prediction and labor force optimization in rail dispatcher scheduling. PhD thesis, Massachusetts Institute of Technology.

Ji, X and K Yao (2017). Uncertain project scheduling problem with resource constraints. *Journal of Intelligent Manufacturing*, 28(3), 575–580.

Kreter, S, A Schutt, PJ Stuckey and J Zimmermann (2018). Mixed-integer linear programming and constraint programming formulations for solving resource availability cost problems. *European Journal of Operational Research*, 266(2), 472–486.

Lambrechts, O, E Demeulemeester and W Herroelen (2008a). Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. *Journal of Scheduling*, 11(2), 121–136.

Lambrechts, O, E Demeulemeester and W Herroelen (2008b). A tabu search procedure for developing robust predictive project schedules. *International Journal of Production Economics*, 112(2), 493–508.

Lambrechts, O, E Demeulemeester and W Herroelen (2011). Time slack-based techniques for robust project scheduling subject to resource uncertainty. *Annals of Operations Research*, 186(1), 443–464.

Li, H and NK Womer (2015). Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. *European Journal of Operational Research*, 246(1), 20–33.

Moradi, M, A Hafezalkotob and V Ghezavati (2019). Robust resource-constrained project scheduling problem of the project's subcontractors in a cooperative environment under uncertainty: Social complex construction case study. *Computers & Industrial Engineering*, 133, 19–28.

Möhring, R (1984). Minimizing costs of resource requirements in project networks subject to a fixed completion time. *Operations Research*, 32, 89–120.

Qi, JJ, YJ Liu, P Jiang and B Guo (2015). Schedule generation scheme for solving multi-mode resource availability cost problem by modified particle swarm optimization. *Journal of Scheduling*, 18(3), 285–298.

Ranjbar, M, F Kianfar and S Shadrokh (2008). Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm. *Applied Mathematics and Computation*, 196, 879–888.

Rodrigues, S and D Yamashita (2010). An exact algorithm for minimizing resource availability costs in project scheduling. *European Journal of Operational Research*, 206, 562–568.

Shadrokh, S and F Kianfar (2007). A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty. *European Journal of Operational Research*, 181, 86–101.

Uysal, F, SK Isleyen and C Çetinkaya (2018). Resource constrained project scheduling with stochastic resources. *Journal of Applied Research on Industrial Engineering*, 5(1), 39–49.

Van den Berg, PL and K Aardal (2015). Time-dependent mexclp with start-up and relocation cost. *European Journal of Operational Research*, 242, 383–389.

Van Peteghem, V and M Vanhoucke (2013). An artificial immune system algorithm for the resource availability cost problem. *Flexible Services and Manufacturing Journal*, 25, 122–144.

Van Peteghem, V and M Vanhoucke (2015). Heuristic methods for the resource availability cost problem. In *Handbook on Project Management and Scheduling Vol.* 1, Chapter 16, C Schwindt and J Zimmermann (Eds.), pp. 339–359. Cham: Springer International Publishing.

Vanhoucke, M, E Demeulemeester and W Herroelen (2003). A random network generator for activity-on-the-node networks. *Journal of Scheduling*, 6, 17–38.

Yamashita, DS, VA Armentano and M Laguna (2007). Robust optimization models for project scheduling with resource availability cost. *Journal of Scheduling*, 10(1), 67–76.

Yamashita, D, V Armentano and M Laguna (2006). Scatter search for project scheduling with resource availability cost. *European Journal of Operational Research*, 169, 623–637.

Biography

Hossein Moghaddaszadeh obtained his BSc in Mechanical and Industrial Engineering from the Sharif University of Technology and MSc in Industrial Engineering from the Ferdowsi University of Mashhad. His research interest includes Operations Research and Scheduling.

Mohammad Ranjbar is Full Professor of Industrial Engineering at the Ferdowsi University of Mashhad. His research interests include Logistics, Transportation and Scheduling. He has published 40 papers in this research area.

Negin Jamili is PhD student at Rotterdam School of Management, the Erasmus University. She received her BSc and MSc in Industrial Engineering from the Ferdowsi University of Mashhad. Her research interests include Supply Chain Management, Warehousing, Scheduling and Transportation and Logistics.