An Evolutionary Approach for Resource Constrained Project Scheduling with Uncertain Changes

Forhad Zaman, Saber Elsayed, Ruhul Sarker, Daryl Essam and Carlos Coello Coello

***Abstract*—The Resource Constrained Project Scheduling Problem (RCPSP) belongs to a class of complex constrained optimization problems. As of the literature, RCPSPs have been widely studied while considering that the activity durations are known with certainty. However, in practice, the durations are usually uncertain for many projects. In this research, the uncertainty in activity durations is represented as stochastic changes. So the new problem is recognized as SRCPSPs. One of the popular approaches for solving SRCPSPs is to use Evolutionary Algorithms (EAs) for scheduling, by applying a suitable number of scenarios for solution evaluations. These scenarios are generated based on the stochastic parameters. In this approach, the fitness evaluation component is computationally very expensive. In this paper, we propose an algorithm, based on two multi-operator based EAs, to deal with the optimization process for scheduling and which also uses a simple strategy to reduce the fitness evaluation costs. A set of test problems with up to 120 activities have been solved using this proposed approach and previous state-of-the-art algorithms. The results obtained by the former were found to be of acceptable quality with a significant reduction of computational time.**

***Index Terms*—RCPSP; uncertainty; evolutionary algorithm; differential evolution; genetic algorithm.**

I. Introduction

A resource constrained project scheduling problem (RCPSP) consists of a number of activities, each of which has a certain duration and limited resources. Its main goal is to determine a schedule for performing the activities, by minimizing the finish time of the last activity, while satisfying any precedence relationships among the activities and for resource availability. This problem has become very popular over the last few decades because of its applicability to various practical problems, including job shop, assembly shop, manufacturing, and construction projects [1].

RCPSP is known to be a NP-hard problem and is computationally expensive [2]. Over the last few decades, numerous approaches for solving RCPSP have been developed, including exact, heuristic, and meta-heuristic, for example, genetic algorithm (GA) [3], greedy randomized adaptive search procedure (GRASP) [4], adaptive robust optimization model [5], local search procedure [6] and heuristic technique [7]. In those studies, the activity durations are mainly considered as known with certainty. However, in many cases in practice, the activity durations are uncertain. RCPSPs with uncertain activity durations are challenging research problems, recognized as stochastic RCPSPs (SRCPSPs) [7].

There are three approaches to deal with SRCPSPs. They are: predictive, proactive and reactive. The predictive approach applies the average value of the durations and solves the problem in a deterministic manner. This approach produces a single schedule which is usually considered as a baseline solution. The proactive approach takes into account the variations in activity durations and determines schedules showing the variations in project completion times. This approach assumes, that the pattern or distribution of uncertainty for activity durations, are known in advance. In a reactive approach, initially a baseline solution is generated by considering deterministic activity durations. It is then re-scheduled when any activity duration is subjected to any uncertainty [6].

In the case of uncertainty, it is expected to provide solutions with the statistical behavior of the project's duration. In the literature, the approaches are broadly categorized as: (i) redundancy-based; (ii) contingent; and (iii) robust or stable scheduling. The first schedules a project while considering additional delay times for its activities to minimize risks, the second produces multiple baseline solutions for a project, and the last obtains a solution through an optimization process, in which the objective function measures robustness, by means of a number of approaches; for example, minimizing the expected fitness values (FVs), taking into account deviations of the FVs under uncertainties, the worst-case values, and solving the problem while considering the softness of the soft constraint [6], [8].

Over the last few decades, many approaches for obtaining stable solutions to SRCPSPs have been developed, for example, robust optimization [9], fuzzy [10], policy-based [11] and scenario-based approach [12]. Among them, the scenario-based approach is the most popular one that has been successfully applied to many uncertain optimization problems, including SRCPSPs [8]. In it, each scenario represents a particular instance of the SRCPSP, by first fixing uncertain parameters, and then solving each instance in a deterministic manner. The expected objective function value is then calculated using the objective function values of the scenarios. For a discrete

The first three authors are with the School of Engineering and Information Technology (SEIT) at the University of New South Wales (UNSW), Canberra at the Australian Defence Force Academy, and the last author is with the *Depto. de Sistema, UAM-Azcapotzalco*, Mexico. Emails: {f.zaman, s.elsayed, r.sarker and d.essam}@unsw.edu.au, and ccoello@cs.cinvestav.mx

uncertain variable, a scenario tree is a common approach for generating such scenarios, where each represents a sample path.

In a general scenario based approach, it is recommended to consider a large number of scenarios in generating solutions. Earlier research showed that, a large number of scenarios need to be generated to cover an uncertain variable's 3σ areas [8]. Such an approach requires a significantly higher computational effort when the parameters are continuous and uncertainty is represented by probability distributions [13]. It is well-known that the scenario dependent approach, combined with population-based algorithms for SRCPSP, is computationally expensive, but produces better solutions than conventional methods [1].

In this paper, to reduce the computational burden of the scenario dependent population-based algorithm in solving SRCPSPs, we investigate the effect of limiting the number of individuals and/or generation, for stochastic fitness evaluation, such as evaluating all individuals in all generations, all individuals in the final generation only, and only the best individual in all generations. In all cases, all generated scenarios are applied and these scenarios are randomly generated based on the uncertain activity durations. For the third option, to find the best individual, all individuals in a given generation are evaluated using a single scenario which is based on the mean values of the uncertain parameters. Then the average fitness value of the best individual is calculated for all scenarios. For experimental study, the scenarios are generated using five different probability distributions. The basic search algorithm consists of two multi-operator evolutionary algorithms (MOEAs), that are a multi-operator GA (MOGA) and multi-operator DE (MODE). Although the same numbers of individuals are initially used for both algorithms, these numbers are updated in each generation, based on their performance in the previous generations. In this case, the better performing algorithm is allowed to evolve for a certain number of generations (defined as a cycle) alone and then the process is restarted with two algorithms. The parameters of MOEAs are selected self-adaptively, based on their performance during the evolutionary process.

The performance of the scenario-based evolutionary approach was evaluated by solving a number of well-known SRCPSPs, with up to 120 non-dummy activities. The parameters of the problems were taken from PSPLIB [14] and the uncertain activity durations were taken from [11]. A comparison of the results obtained by the proposed approach and several state-of-the-art algorithms, reveals that the proposed algorithm, with the third option, outperforms the others in terms of solution quality, with a significant reduction in computational time. This research uncovers the fact, that most current methodologies are using more computation than required.

The contributions of this paper can be summarized as: (i) SRCPSPs are defined as a scenario-based model, that considers the uncertain activity duration can be either real or integer; (ii) different options of fitness evaluations using scenarios are investigated, in order to reduce computational time; (iii) the elite solution is determined, based on the expected fitness values of the best individuals from both the parents and offspring; (iv) a new algorithm was developed to solve this scenario-based model; and (v) the results obtained are compared with those of many existing algorithms, which shows that the proposed algorithm is superior.

The structure of this paper is as follows: Section II describes the mathematical modeling of the scenario-based SRCPSPs and provides a relevant literature review; Section III discusses the proposed solution approach; Section IV illustrates the computational results and statistical comparisons with state-of-the-art algorithms; and Section V presents the conclusion and suggested future work.

II. Background of SRCPSP

In this section, the formulation of SRCPSPs and their existing solution approaches in the literature are presented.

A. Problem description

The aim of a SRCPSP is to schedule a project with a stable solution in an uncertain environment. Generally, a project network is used to explain the logical relationships between the activities in a given project; for example, Fig. 3 shows a project with 11 activities, in which the node set ($D = \{1, 2, \ldots, 11\}$) represents the activities, the arc sets (A) their start and finish times (ST and FT, respectively), and the zero lag of their precedence relationships. For each project, there are two dummy activities, 0 and $D+2$, which indicate the *start* and *finish* times of the project, respectively. Therefore, a project consists of $D+2$ activities and each cannot be started until all its precedence activities are completed. It is assumed, that if an activity starts, it cannot be stopped until it's entirely finished, also that the duration of each activity (d) is uncertain and denoted as $\tilde{d}$ [1], [15].

The objective of a SRCPSP, is to minimize the overall project duration, which is called the make-span of a project. Its constraints are of satisfying all the precedence relationships and resource capacity limits, which are mathematically expressed as:

$$\text{Minimize: } E\left[FT_{D+2}\right] = \frac{1}{N_S}\sum_{s=1}^{N_S} FT_{(D+2),s} \tag{1}$$

Subject to:

$$FT_{1,s} = 0,\ s \in N_S \tag{2}$$

$$FT_{i,s} \leq FT_{j,s} - \tilde{d}_{s,j}, \forall\,(j,i) \in A, s \in N_S \tag{3}$$

$$\sum_{j \in A_t} r_{jk} \leq R_k,\ \forall j, t, k, s \tag{4}$$

$$FT_{j,s} \geq 0, \forall j \tag{5}$$

where $E\left[FT_{D+2}\right]$ is the expected FT of the *finish* dummy activity, calculated by averaging FT_{D+2} over N_S scenarios, $FT_{(D+2),s}$, the FT of the *finish* dummy activity in the s^{th} scenario and $FT_{1,s}$, that of the *start* dummy activity which is always zero. Constraint (3) is used to ensure that the i^{th} activity cannot be started until all its predecessors have finished, with $\tilde{d}_{s,j}$ indicating the duration of the j^{th} activity in the s^{th} scenario, that is randomly generated using a PD with a mean of (details are provided in subsection III-B1). Constraint (4) indicates that activity j requires a k^{th} type of r_{jk} resource, which must not be greater than its maximum limit of R_k. As there are K types of renewable resources in a project, they are limited to being used at a certain time, but can be re-used later, for example, machines and manpower. The symbol A_t represents a set of activities running during a time period, $t \leq FT_{D+2}$, with constraint (5) is used to ensure that an activity's FT is not negative.

B. Relevant literature review

Over the last few decades, deterministic RCPSPs have been studied extensively. Since they are NP-hard problems, numerous approaches have been developed to obtain better solutions. Of them, meta-heuristic algorithms have performed well, due to their flexible characteristics which avoid local minima [16]; for example, Alcaraz et al. [17] developed a robust GA for scheduling different projects, which although performing well, is computationally expensive. To reduce time, Debels et al. [18] proposed a decomposition-based GA and Valls et al. [19] a hybrid GA, in which the search operators are modified and a local search introduced to enhance the solution quality. Using a local search, other population-based algorithms perform well, for example, differential evolution (DE) [20], particle swarm optimization (PSO) [21] and DE with fuzzy clustering [10]. Based on the results obtained by these algorithms, it is evident that some are superior for certain problems, but, are inferior for others, which is true even for a single problem. It is found that an algorithm may perform well in an early stage of evolution but become stuck in local optima later. To overcome this drawback, Saber et al. [1] developed a multi-method based algorithmic framework called COA, which consists of two algorithms: a MOGA and MODE, each of which uses multiple search operators, with the results obtained being better than those in the existing literature. However, these methods were designed for deterministic RCPSPs, in which the durations of activities are considered to be known discrete values, whereas because in practice they are uncertain and may change for many reasons, including unavailable resources, bad weather, delayed delivery and lack of workers, considering them known may result in a project's schedule performing poorly.

Recently, many researchers considered RCPSPs with uncertain durations, called SRCPSPs, in which an activity's duration is not assumed to be known in advance, but are represented by a random vector or scenario as $\tilde{d} = \{d_1, d_2, \ldots, d_{N_S}\}$, where $d_s, s \in N_S$ is the s^{th} random scenario of a duration with a known PD [11]. However, this assumes that the distributions of the durations of the activities are discrete, which results in the SRCPSP being a generalization of a deterministic RCPSP. However, due to the uncertain behavior of activities' durations, the solution to a SRCPSP may not be a single schedule, but multiple ones depending on the scenarios of an activity's duration. In other words, since the finish time of an activity can be a multiple, depending on its duration scenarios, the start time of its successor must also be a multiple. Consequently, a number of solutions are found for a SRCPSP, with some researchers calling them policies [3]. A policy or strategy is defined as updating a solution on an on-line basis, that determines which activities are to be started at a certain decision time (t), based on a given PD and prior knowledge up to $t-1$.

Several classes of such policies can be found in the literature, for example, earliest-start $\left(\pi^{ES}\right)$, pre-processor $\left(\pi^{PR}\right)$, pre-selective $\left(\pi^{PS}\right)$, linear pre-selective $\left(\pi^{LPS}\right)$, resource-based $\left(\pi^{RB}\right)$, and activity-based $\left(\pi^{AB}\right)$ [13], with many researchers using them to solve SRCPSPs; for example, Ballestï¿œn et. al [3] and Francisco et. al. [22] used a π^{AB} policy, Ashtiani et. al [23] used a π^{PR} and Rostami et. al. [11] used different generalized processor policies. Also, Fernandez et al. [24] solved a SRCPSP using a scheduling policy represented as a multi-stage stochastic optimization problem. Recently, a closed-loop policy for SRCPSPs with a dynamic programming approach was introduced by Li and Womer [2], that outperformed the above-mentioned open-loop policies. However, it was also found that closed-loop policies are only superior for a SRCPSP with a asymmetric PD of durations, while open-loop ones are better for other cases. Considering either an open- or closed-loop policy, particularly RB, AB or PR, is computationally intractable for a large practical SRCPSP, as the set of scenarios increases exponentially, according to the number of activities [11].

Another useful approach, in which a SRCPSP is represented as a scenario-based model and the expected objective function is evaluated by considering many possible scenarios of activities' durations, is found in the literature [15]. Its main advantage is that the operator does not need to know the actual values of the uncertain parameters in advance. Instead, many sets of random values are generated for every uncertain parameter using a known PD and then their FVs are independently evaluated and their average value is considered the expected FV of a given schedule. Over the last few years, this scenario-based method has become a popular choice for many optimization problems with uncertainties, such as project scheduling [9], electrical generator planning [8] and logistics [25]. However, this approach is generally computational expensive. To reduce computational time,

Tseng et al. [12] developed a scenario-based SRCPSP model containing a proposed technique for reducing the number of scenarios, Zaman et. al [15] solved a SRCPSP using an evolutionary algorithm (EA), with the scenarios of activity durations being considered only in the final generation, and their known values were used in other generations. Although this approach reduced computational time significantly, the schedule obtained by that approach may not be optimal, as after applying the scenarios, the best schedule was formed among the final population members and so had no chance of further improvement. In addition, the scenarios of the durations were rounded to their nearest integer values, that caused it to loss a PD's property. To best of our knowledge, solving a continuous scenarios based SRCPSP using EAs, has not yet been explored.

III. Proposed Approach

In this paper, the SRCPSP is modeled as a scenario driven problem with continuous decision variables. The original COA was designed for solving deterministic RCPSP, where the activity durations were considered as integers [1]. As the uncertain durations are continuous, we modified the algorithms to deal with continuous variables. The uncertain variables of durations are represented by a number of scenarios (N_S) generated using various PDs, as shown in subsection III-B, with the algorithm called a 'scenario-based COA' ('S-COA').

S-COA is a population-based method which starts with an initial population of N_P random individuals, as shown in subsection III-A, with new individuals always used in a serial schedule generation scheme (SSGS) to obtain their feasible schedules and a local search is applied to enhance the quality of the schedules. Firstly, N_P individuals are evaluated considering their deterministic duration values, using Eqns. (1) to (5). Then, based on their FVs, the best individual is selected and its expected FV (EFV) is evaluated for N_S duration scenarios, as shown in subsection III-B. Note that a random duration from N_S scenarios can be a continuous or discrete value, with the process for dealing with a continuous one in S-COA, as described in subsection III-B.

In subsequent generations, new individuals are generated using one or both of the S-COA algorithms, MOGA and MODE, based on their prior performances for generating better individuals than their parents, as discussed in subsection III-C. After their generation, N_P new offspring are evaluated using their deterministic durations and based on the minimum finishing times of the last dummy activity, the best individual is selected. Then, the EFVs of the new best and its parent's best, are evaluated under the same N_S duration scenarios and the best individual is merged with its selected offspring. This process continues until the algorithm reaches its maximum fitness function evaluations (MFFEs). The pseudo-code of the proposed S-COA is presented in Algorithm 1 and its details are discussed in the following subsections.

Algorithm 1 Pseudo-code of solution approach

Require: Require: N_S, $MFFE$ and N_P.
1: Generate an initial population of size N_P, as discussed in subsection III-A.
2: Generate feasible schedules and apply a local search to all N_P individuals, and evaluate them, as discussed in subsection III-B.
3: Evaluate the expected FV (EFV) of the best individual for N_S scenarios, as shown in subsection III-B.
4: Set the current FFE (cfe) to, $cfe = N_P + N_S$.
5: **while** $cfe \leq MFFE$ **do** ▷ *Main loop*
6: Generate N_P new offspring based on the MOGA and MODE operators, as discussed in the optimization methods in subsection III-C.
7: Evaluate the FVs of the new offspring as discussed in step 2 . Set, $cfe = cfe + N_P$.
8: Based on the best FVs, find the best individual, apply N_S scenarios to the new and old best individuals, and evaluate their EFVs, and set, $cfe = cfe + 2 * N_S$.
9: Based on these EFVs, select the best individual for this iteration.
10: **end while**

A. Representation and initial generation

The decision variables of SRCPSP are sequences of activities, generated using random permutations as:

$$\vec{x_i} \in perm\binom{\Delta}{\Delta}\ \Delta = \{1, 2, \ldots, D-1\}\,,\ \forall i \in N_P \qquad (6)$$

where $\vec{x_i}$ is the i^{th} individual in the N_P population, $perm\binom{\Delta}{\Delta}$ the permutation or random combination of the non-dummy activities. The number of decision variables depends on the number of activities in a project i.e., $D+2$.

B. Fitness evaluation with uncertainty

As a random or newly generated child solution may not be feasible for satisfying its precedence and resource constraints, a SSGS is employed to make it feasible. In it, a new solution $(\vec{x_i}, i \in N_P)$ is decoded by selecting the activities in an order based on their earliest starting times subject to their satisfying the precedence and resource constraints. To obtain an entire feasible schedule, $\sum_j d_j$ iterations are required [11]. However, this may not be valid when the duration of an activity is considered a random value which can be continuous. To deal with continuous values, SSGS firstly rounds the solution to a given number of decimal points (n); for example, if $n = 10$ and $d = \{5.265, 3.253, 1.265, 1.021\}$, the rounded durations are $d = round\,[(d * n)/n]$ which is equivalent to $d = \{5.30, 3.30, 1.30, 1.00\}$. Therefore, to obtain a complete feasible schedule, $\sum_j nd_j$ iterations are required. Although higher values of n produce more accurate solutions, they increase the computational time significantly. In SSGS, an appropriate activity is selected in each iteration and is inserted into a partial schedule, based on its earliest

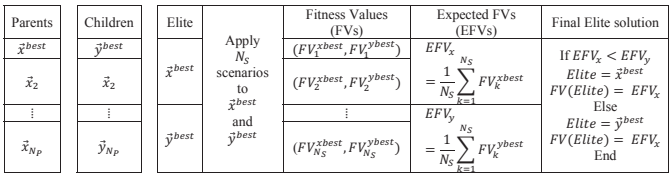

Fig. 1: Graphical process for selecting the elite member

possible time subject to its precedence and resource constraints. During this insertion, the starting times of the activities already scheduled remain unchanged, while the finishing time of each is recorded.

To enhance the FVs of individuals, a local search, in which their activities are sorted in descending order of their finishing times, is employed [1]. Then, the activities are pushed forward as far as possible, while maintaining the precedence and resource constraints while those of the updated individuals are sorted again, based on their starting times. Finally, the updated solutions are used by SSGS to generate feasible schedules. Subsequently, the individuals are sorted based on their finishing times to select an elite one. Two elite individuals are selected (or one if it is in the initial generation), one from the parents (x^{best}) and another from their children (y^{best}) and they are evaluated with N_S scenarios generated using a PD, as discussed in subsection III-B1. Once N_S scenarios are evaluated, the EFVs of both elite solutions are determined by averaging their FVs for all scenarios. Based on the EFVs of the two elite members, the best one is selected for the next generation. Fig. 1 shows a graphical overview of the selection of one of two elite individuals, based on their EFVs.

Once the final elite member is selected, it is merged with other individuals already selected, based on their deterministic FVs, for the operations of the optimization methods MOGA and MODE, as discussed in subsections III-C1 and III-C2 for MOGA and MODE, respectively.

1) Scenario generation: In this paper, the durations of activities are assumed to be uncertain and follow a particular PD. As when a project's activities are scheduled, their durations are considered random values of N_S scenarios. We generate them for each activity's duration using three well-known continuous PDs that are common in SRCPSPs [11], [16]. They are discussed as follow.

a) Uniform Distribution: It is a common type of PD that has a constant probability that can be discrete or continuous, we assume a continuous one, with N_S scenarios of a given duration generated as:

$$\tilde{d}_{s,j} = lb_j + (ub_j - lb_j) rand_s \tag{7}$$

$$\forall s = 1, 2, \ldots, N_S;\ \forall j \in D + 1$$

where $\tilde{d}_{s,j}$ is a random value of the j^{th} activity's duration in the s^{th} scenario, $rand_s \in [0, 1]$ is a random value between 0 and 1, and lb_j and ub_j are the lower and upper bounds of the j^{th} activity, respectively.

b) Exponential Distribution: This distribution is usually modeled with a constant rate of failure and a larger standard deviation (Std) than the uniform one. The N_S random scenarios of a given duration, using an exponential distribution, are generated as:

$$\tilde{d}_{s,j} = Exp(\mu_j), \forall j \in D + 1; \forall s = 1, 2, \ldots, N_S \tag{8}$$

where μ_j is the mean or deterministic value of the j^{th} activity's duration.

c) Beta Distribution: This distribution is very well known for project scheduling with uncertainty. Many RCPSPs have been modeled that uses it with two shape parameters (α and β) to generate a random value between 0 and 1, with N_S scenarios of the activity's duration generated as:

$$\tilde{d}_{s,j} = lb_j + (ub_j - lb_j) Beta_s(\alpha, \beta) \tag{9}$$

$$\forall s = 1, 2, \ldots, N_S;\ \forall j \in D + 1$$

where $Beta_s(\alpha, \beta)$ is a random value between 0 and 1 in the s^{th} scenario, generated by using a beta distribution with parameters, α and β.

C. Optimization methods

In this research, two algorithms, MOGA (Alg_1) and MODE (Alg_2), are used in a single framework in which both are performed sequentially one after another. Initially, their probabilities ($prob_1$ and $prob_2$) are set to 1 and in each generation, two random numbers are generated as $rand_i \in [0, 1], i = 1, 2$. Then, if $rand_i \leq prob_i$, a new solution is generated using Alg_i. After a certain number of generations, called a cycle (CS), $prob_i$ is updated based on its performance in previous generations as [1]:

$$prob_i = \max\left(0.1, \min\left(0.9, \frac{\sum_{g=1}^{CS} SR_{i,g}}{\sum_{i=1}^{2}\sum_{g=1}^{CS} SR_{i,g}}\right)\right) \tag{10}$$

where g is the current generation number, and $SR_{i,g}$ the success rate of the i^{th} algorithm in the g^{th} generation, determined as:

$$SR_{i,g} = \begin{cases} \frac{f^{best}_{old,i,g} - f^{best}_{new,i,g}}{f^{best}_{old,i,g}} & f^{best}_{old,i,g} \neq f^{best}_{old,i,g} \\ 1 & f^{best}_{old,i,g} = f^{best}_{old,i,g} \end{cases} \tag{11}$$

where $f^{best}_{old,i,g}$ and $f^{best}_{new,i,g}$ are the best FVs of the parents and offspring of the i^{th} algorithm in the g^{th} generation, respectively, $\forall i \in \{1, 2\}$, and $g \in N_P$.

As S-COA considers two MOEAs, for MOGA an integer-based crossover and mutation operator are employed, while for MODE, two mutation operators, DE_1 and DE_2, are used with a crossover, as discussed in the following subsections.

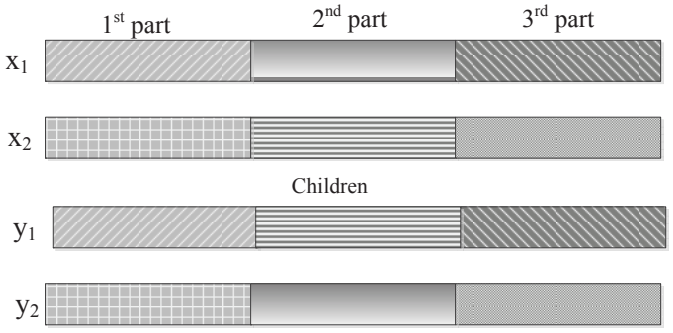

Fig. 2: Two-point crossover

1) MOGA: In MOGA, two crossovers, (i) a two-point, and (ii) a uniform, with a left-shift mutation operator is used. The number of individuals assigned to each operator, depends on their performance in previous generations, as calculated as:

$$nI_1 = \min\left(N_P - 1, \left(\max\left(1, \sum_{k=1}^{N_P} rand_1 \leq Prob_{op1}\right)\right)\right) \tag{12}$$

$$nI_2 = N_P - nI_1 \tag{13}$$

where $rand_1 \in [0,1]$ and $Prob_{op1}$ is the probability of the two-point crossover, which is initially set to 0.5, and subsequently updated using Eqn. (15). The symbols nI_1 and nI_2 indicate the numbers of individuals to be generated using two-point and uniform crossovers, respectively.

To generate two new individuals using the two-point crossover, firstly, two parents ($\vec{x}_1$ and $\vec{x}_2$) are randomly selected using a tournament pool, next they are divided into three parts and then they combined by exchanging the 2nd part between the two parents while ensuring that there are no redundant elements in any child. Fig. 2 shows the process for generating two children from two parents.

Using the uniform crossover, a child ($\vec{y}_1$) is generated from two random parents $\vec{x}_1$ and $\vec{x}_2$, as:

$$\vec{y}_1 = \begin{cases} x_{1,j} & rand_j \in [0,1] \leq 0.5 \\ x_{2,j} & rand_j > 0.5 \cap x_{2,j} \notin \vec{y}_1 \end{cases}, \forall j = 0, 1, \ldots, D+1 \tag{14}$$

Once a child is generated, a left-shift mutation is used, in which a gene of a child ($y_{i,j} \forall i \in N_P, \forall j \in D+1$) is randomly shifted to its left if $rand_j \in [0,1] \leq MR$, where MR is the mutation rate.

Once all the children are generated, their feasible schedules are obtained, as discussed in subsection III-A. Then, after applying a local search, their FVs are evaluated considering N_S scenarios, as in subsection III-B. To enhance the convergence rate, the elite solutions from the previous generation are merged with the new individuals and $Prob_{op1}$ updated as:

Table I: Example of obtaining a continuous individual from a discrete one

$\vec{x}_i,\ i \in N_P$	1	5	4	2	3	6
$loc(x_i)$	0	3	4	2	1	5
rand$\in [0,1]$	0.32	0.84	0.85	0.04	0.61	0.39
$\vec{x}_i^{cont}$	0.32	3.84	4.85	2.04	1.61	5.39

$$Prob_{op1} = \max\left(0.1, \min\left(0.9, \frac{I_{2p}}{I_{2p} + I_u}\right)\right) \tag{15}$$

where I_{2p} and I_u are the average improvements in the two-point and uniform crossovers, respectively, which are calculated as: $\left(\sum_{i=1}^{N_P} \max\left(0, f_i^{new} - f_i^{old}\right)\right)/N_P, \forall N_P$. Note that when both I_{2p} and I_u are zero, the value of $Prob_{op1}$ is set to 0.5.

2) MODE: Depending on the value of $prob_2$, a number of new individuals are generated, using the operators of MODE. Similar to MOGA, the number of individuals in each operator is determined, based on Eqns. (12) and (13), in which nI_1 and nI_2 represent DE_1 and DE_2, respectively.

In MODE, the real-value individuals, i.e., $\vec{x}_i^{cont}$, are used to generate new individuals ($\vec{y}_i^{cont}$), with the integer individuals first encoded to their continuous ones, which is obtained from a discrete individual (*i.e.*, $\vec{x}_i$). An encoding approach is employed, in which each value of $\vec{x}_i^{cont}$ is encoded, based on its location (loc) in the vector of $\vec{x}_i$, plus a random value between 0 and 1. For example, Table I shows the process whereby a continuous decision vector is obtained from its discrete one. The second row indicates the location of each cell, for example, the second cell is the location of '2' which is 3 in the first row of $\vec{x}_i$. Note that the first and last activities of each project are dummies.

Once $\vec{x}_i^{cont}$ is obtained, the nI_1 number of offspring is generated using DE_1, which is based on '*current-to-rand/bin with archive*', as:

$$y_{i,j}^{cont} = \begin{cases} x_{i,j}^{cont} + F_i\left(x_{r1,j}^{cont} - x_{i,j}^{cont} + x_{r2,j}^{cont} - \widehat{X}_{r3,j}^{cont}\right) \\ \qquad \text{if } rand \leq cr_i \cup j = j_{rand} \\ x_{i,j}^{cont} \qquad \text{otherwise} \end{cases} \tag{16}$$

where, $r_1 \neq r_2 \neq r_3 \neq i \in N_P$ are randomly selected from x^{cont}, and F_i and cr_i are the amplification factor and crossover rate of the i^{th} individual, respectively. As the appropriate selection of these two parameters (F_i and cr_i) is challenging, as it is very important to obtain the best solution to an optimization problem [26], they are self-adaptively adjusted in each generation during the solution process. Details of the self-adaptive mechanism can be found in [1]. The notation *cont* represents the union of the storage archive of all the old individuals and the entire x^{cont}. Note that as this archive is initially empty, but after a certain stage, may become too large, we set a threshold value. If the size of the archive exceeds this threshold, the old values are randomly replaced with new ones.

Table II: Example of decoding to obtain a discrete individual from a continuous one

$\vec{x}_i^{cont}$	0.32	3.84	4.85	2.04	1.61	5.39
$sort\left(\vec{x}_i^{cont}\right)$	0.32	1.61	2.04	3.84	4.85	5.39
Rank= $\vec{x}_i$	1	5	4	2	3	6

Once the nI_1 number of offspring is generated using DE_1, the remaining offspring (i.e., nI_2) are generated using DE_2 which is based on '*current-to-rand/bin without archive*' as:

$$y_{i,j}^{cont} = \begin{cases} x_{i,j}^{cont} + F_i\left(x_{r1,j}^{cont} - x_{i,j}^{cont} + x_{r2,j}^{cont} - x_{r4,j}^{cont}\right) \\ \qquad \text{if } rand \leq cr_i \cup j = j_{rand} \\ x_{i,j}^{cont} \qquad \text{otherwise} \end{cases} \tag{17}$$

where $rand \in [0,1]$, $r_4 \neq r_1 \neq r_2 \neq r_3 \neq i \in N_P$ and j_{rand} is a random gene of $\vec{x}_i^{cont}$.

After generating N_P offspring using DE_1 and DE_2, the continuous-valued $y_i^{cont} \forall i$ are decoded to their integer-valued $y_i \forall i$ by firstly sorting $\vec{y}_i^{cont}$ in ascending order and then producing $\vec{y}_i$ based on its rankings. For example, Table II shows a sample of a discrete schedule obtained from a real-valued one.

After obtaining $\vec{y}_i \forall i \in N_P$, feasible schedules are generated from $y_i \forall i$, as described in subsection III-A. Then, a local search is applied to all the offspring and their FVs evaluated for N_S scenarios, as discussed in subsection III-B. In the selection, the better individuals are selected, based on their corresponding FVs, as:

$$x_{i,j}^{new} = \begin{cases} y_{i,j} & FV(y_{i,j}) \leq FV(x_{i,j}) \\ x_{i,j} & FV(x_{i,j}) < FV(y_{i,j}) \end{cases} \tag{18}$$

where $x_{i,j}^{new} \forall i \in N_P$, $j = 0, 1, \ldots, D$ is the selected j^{th} gene in the i^{th} individual. Finally, the $Prob_{op1}$ for MODE is updated in a similar way to that for MOGA using Eqn. (15).

IV. Computational Results

In this section, several experiments that were carried out to determine the performance of S-COA for solving different SRCPSPs are discussed, and the results obtained are compared with those from state-of-the-art algorithms. For fair comparisons, the SRCPSP benchmarks are taken from the literature. The algorithm is evaluated on problem sets of J30, J60 and J120 activities, all of which are standard data sets for project scheduling problems, containing 400, 600 and 600 RCPSP instances with 30, 60 and 120 non-dummy activities, respectively [1]. All the test problems are solved using S-COA while considering the following N_S scenarios for the activities' durations:

- ***var1***: scenarios are considered for evaluating all individuals in all generations;
- ***var2***: scenarios are considered for evaluating all individuals only in the final generation; and
- ***var3***: scenarios are considered for evaluating the best individual in all generations.

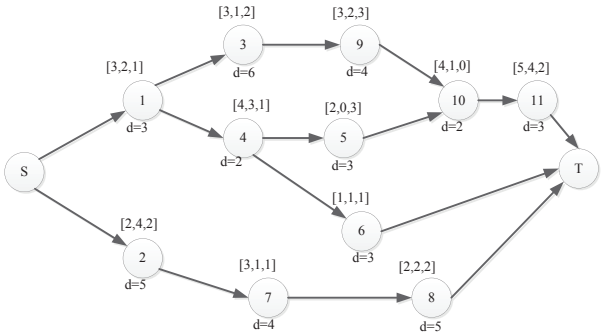

Fig. 3: Small test problem with 11 activities

These three variants of the proposed algorithm (S-COA) are used to solve all the problems to determine the explicit performances of the scenario-based algorithm. Note that *var1* is a popular choice in an evolutionary uncertain environment [8] while *var2* and *var3* are tested in this paper.

For fair comparisons, the MFFEs are tested for both 5000 and 25000 schedules for all the problems, the values of N_S is 100, 50 and 10 for the J30, J60 and J120 ones, respectively, the population size (N_P) to 10 and the other parameters kept the same as in [1]. As we use continuous PDs to generate scenarios of activities' durations, the random values of these durations can be continuous, and to obtain their decimal values to the tenth place, we set $n = 10$ (subsection III-B).

Each test case is run 15 times, with the median values reported for all the variants of the algorithm, until they reach the MFFEs. They are implemented in a Matlab (R2018a) environment on a desktop computer with a 3.4 GHZ Intel Core i7 processor and 16 GB of RAM.

A. Illustrated Problem: Discussion

Initially, we considered a small test problem with 11 non-dummy activities, to determine the performance of the different variants of the proposed algorithm. Its parameters are shown in Fig. 3, with the availability of three types of resources that are A=6, B=7 and C=6 [27].

The optimal solution of this problem is 20 [27] which is easily obtained using the deterministic version of our algorithm. The stochastic nature of this problem is established by considering that the durations are uncertain and follow a beta distribution, with the parameters based on three estimations, *i.e.*, optimistic (a), most likely (m), and pessimistic (b). The α and β parameters for the beta distribution are calculated as:

$$\phi_j = \frac{5a_j - 4m_j - b_j}{a_j + 4m_j - 5b_j} \; \forall j = 1, 2, \ldots, 11 \tag{19}$$

$$\alpha_j = \phi_j \beta_j \; \forall j = 1, 2, \ldots, 11 \tag{20}$$

$$\beta_j = \frac{-(\phi_j^2 - 34\phi_j + 1)}{(\phi_j + 1)^3} \; \forall j = 1, 2, \ldots, 11 \tag{21}$$

Table III: Results for the small test problem obtained by S-COA

Alg.	Make-span			Time
	Min.	Mean	Max.	(sec.)
var1	21.30	21.39	21.50	3.62
var2	21.40	21.72	23.90	6.07
var3	21.30	21.39	21.50	2.24

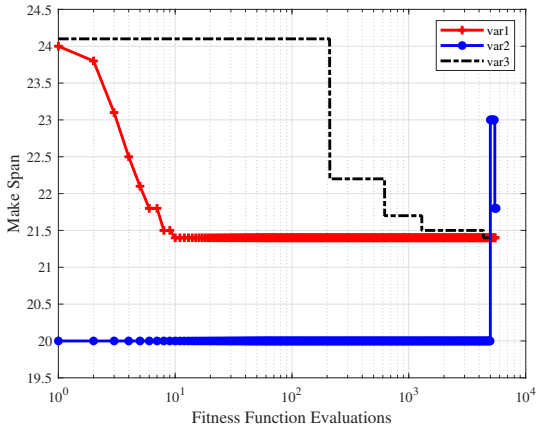

Fig. 4: Convergence plots of the small test problem

where $a_j = 0.8d_j$, $m_j = d_j$ and $b_j = 1.5d_j \, \forall j$, and d_j is the duration of the j^{th} activity. Given a sample scenario size of 100, the best solution obtained to this problem is 21.706 using the tabu search algorithm [27]. We also solved the same problem, using the above three different variants of the proposed algorithm.

With the MFFEs set to 5k, each variant of the algorithm was run 15 times and its mean value is reported in Table III. It can be seen that the results for the project durations using *var1* and *var3* are the same, while the average computational time for *var1* is greater than that for *var2* and *var3*.

Fig. 4 shows convergence plots of the three variants of the proposed algorithm. It can be seen that *var1* and *var3* obtain the same best solutions, while *var2* solves the problem in a deterministic way until the final generation and then considers scenarios which increase the make-span in comparison with those of previous generations. However, as the best solution obtained by *var2* is worse than those from *var1* and *var3*, based on the quality of solutions and computational times, it can be easily concluded that *var3* is the best option for solving such a scenario-based SRCPSP. Therefore, for the following SRCPSPs, we compare only the results from *var3* with those from the state-of-the-art algorithms.

B. Experimental setup for PSPLIB benchmarks

In this section, the performance of the proposed S-COA is tested by solving the well-known PSPLIB benchmarks with J30, J60 and J120 instances, with the difficulty of each defined by the three parameters of network complexity (NC), resource factor (RF) and resource strength (RS). NC indicates the average number of predecessors in each activity, RF the average percentage of various types of resource usage per non-dummy activity which has a non-zero duration and RS the availability of resources. For a fair comparison, the values of NC, RF and RS for the J30 and J60 instances are set to $NC \in \{1.5, 1.8, 2.1\}$, $RF \in \{0.25, 0.5, 0.75, 1\}$ and $RS \in \{0.2, 0.5, 0.7, 1\}$, and those for the J120 ones $NC \in \{1.5, 1.8, 2.1\}$, $RF \in \{0.25, 0.5, 0.75, 1\}$ and $RS \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ [1].

Table IV: Parameters of PD

PD	Limits		Mean	Variance
	lb	ub	μ	σ
U1	$d_j - \sqrt{d_j}$	$d_j + \sqrt{d_j}$	-	$d_j/3$
U2	0	$2d_j$	-	$d_j^2/3$
Exp	-	-	d_j	d_j^2
B1	$d_j/2$	$2d_j$	-	$d_j/3$
B2	$d_j/2$	$2d_j$	-	$d_j^2/3$

After 15 independent runs of each test case, the median results are compared with those available from state-of-the-art algorithms for SRCPSPs. Firstly, we consider a GA [3] and GRASP [22], both of which use an activity-based (AB) policy and are called AB-GA and AB-GR, respectively. Secondly, a two-phase GA [23] called PPGA and an approximate dynamic programming (ADP) algorithm based on a hybrid look-back and look-ahead (HBA) approximation architecture [2], called ADP-GBA, are considered. Finally, we compare our algorithms with the recently proposed different generalized processor policies (GP-policies) for SRCPSPs [11]. Also, the results obtained by several approaches for J120 activities found in the literature, such as an estimation of distribution algorithm with RB (RB-EDA) [13] and a two-phase meta-heuristic consisting of GRASP and a GA with GP (GP-H) [11], are compared with those obtained by our algorithms.

We consider five types of PD, Uniform1 (U1), Uniform2 (U2), Exponential (Exp), Beta1 (B1) and Beta2 (B2), with their lower and upper bounds (lb and ub, respectively), means (μ) and variances (σ) shown in Table IV. The parameters α and β for the B1 and B2 distributions are set as $\alpha = \frac{d_j}{3} - \frac{1}{3}$, $\beta = d_j - \frac{2}{3}$, and $\alpha = \frac{1}{6}$, $\beta = \frac{1}{3}$, respectively. It is clear in Table IV, that the durations with the Exp distribution have a maximum σ, followed by those with the U2 and B2 ones, and then the U1 and B1 ones.

C. Experimental results

The average results after solving each test problem up to 5k and 25k MFFEs, using the proposed S-COA with *var3*, are presented in Tables V, VI and VII for the J30, J60 and J120 instances, respectively, with the notation 'NR' indicating that these values are not found in the literature. The column labeled 'Gap', indicates the average percentage deviation of the expected make-span from the critical path length (CPL), with the deterministic durations [13]:

Table V: Comparison of results obtained by S-COA and state-of-the-art algorithms for J30 instances

Algorithm	MFFE	Gap				
		U1	U2	Exp	B1	B2
PPGA [23]	5k	19.87	30.67	45.56	19.93	30.76
	25k	19.59	30.24	46.12	19.49	30.22
ADP-HBA [2]	NR	16.63	42.37	45.13	12.60	16.63
LFT [28]		21.6	30.89	46.47	21.59	30.87
SLFT [28]	25k	21.6	30.83	46.32	21.6	30.76
DH [28]		21.36	31.18	46.86	21.36	31.21
S-COA	5k	1.56	8.67	16.66	1.29	7.72
	25k	**0.83**	**7.26**	**13.79**	**0.74**	**6.14**

Table VI: Comparison of results obtained by S-COA and state-of-the-art algorithms for J60 instances

Algorithm	MFFE	Probability distribution				
		U1	U2	Exp	B1	B2
PPGA [23]	5k	18.91	29.08	45.74	18.98	29.17
	25k	18.32	28.57	45.36	18.31	28.77
ADP-HBA [2]	NR	14.14	38.32	39.76	**10.46**	**16.84**
LFT [28]	NR	19.94	28.49	44.97	19.95	28.63
SLFT [28]	NR	19.89	28.42	44.94	19.90	28.55
S-COA	5k	12.76	19.31	28.70	12.62	18.54
	25k	**11.83**	**18.05**	**26.38**	11.79	16.92

$$\text{Gap} = \frac{1}{R}\sum_{r=1}^{R}\frac{E_r(f) - CPL_r}{CPL_r} \tag{22}$$

where $E_r(f)$ is the expected make-span of the r^{th} problem, which is determined by the means of simulations with N_S scenarios with values of R are 400, 600 and 600 for the J30, J60 and J120 instances, respectively.

In Tables V to VII, it can be seen that the proposed S-COA algorithm, consistently obtains better results for both the 5k and 25k cases, than the state-of-the-art algorithms. However, the maximum gaps are in the Exp distribution, followed by U2 and B2, and the minimums in U1 and B1, which is consistent with the assumptions of the parameters for different PDs in Table IV.

Table VIII presents the average computational times in seconds for different instances. As the state-of-the-art algorithms are not implemented in our platform, it would not be fair to compare their reported CPU times with ours. However, it can be seen that the CPU times of the proposed algorithm significantly increase when the MFFEs are changed to 25k from 5k.

D. Uncertainty and PD

As already noted, the expected make-span varies widely when the pattern of the uncertainty of a project's durations is changed, with the changes possibly reflected in the σ of the solutions obtained after applying N_S scenarios to the best solution. In other words, a small value of N_S indicates that the solution is more stable, even after applying the scenarios.

Table IX presents the σ values of the three variants (*var1, var2 and var3*) of the proposed S-COA for different problems with different PDs. It can be seen that the larger values are in the Exp distribution for all cases, obviously as they have the maximum Std, while the smaller values are in the U1 and B1 ones, as they have the minimum Std. Comparing these values obtained from three variants of the algorithm, it is clear that they are almost the same for all cases, which indicates that regardless of whether an algorithm considers scenarios in the final generation (*var2*) or applies scenarios to the best individual in all generations (*var3*), it covers the same level of uncertainties, as when the scenarios are considered for all the individuals in every generation (*var1*). Therefore, it could be said that how the scenarios are used in any stage of the solution process does not affect the consideration of uncertainties.

Table VII: Comparison of results obtained by S-COA and state-of-the-art algorithms for J120 instances

Algorithm	MFFE	Probability distribution				
		U1	U2	Exp	B1	B2
AB-GA [22]	5k	51.49	78.65	120.22	-	-
	25k	49.63	75.38	116.83	-	-
AB-GR [22]	5k	46.84	72.58	114.42	47.17	75.97
	25k	45.21	70.95	112.37	45.60	74.17
RB-EDA [13]	5k	47.29	59.54	72.50	47.65	58.29
	25k	46.66	56.07	72.05	47.04	57.82
GP-H [11]	5k	46.71	55.95	71.71	46.87	55.95
	25k	44.98	55.37	71.29	45.12	55.42
PPGA [23]	5k	48.86	59.91	76.03	49.01	58.82
	25k	47.21	58.07	74.56	47.25	57.95
EDA [13]	5k	47.29	56.54	72.50	47.65	58.29
	25k	46.66	56.07	72.05	47.04	57.82
ADP-HBA [2]	NR	42.11	71.94	74.90	38.93	45.39
RB-LFT [28]	25k	48.05	55.59	70.95	48.05	55.56
RB-SLFT [28]	25k	48.04	55.48	70.76	48.04	55.45
S-COA	5k	34.57	37.15	39.58	34.53	35.40
	25k	**33.30**	**34.87**	**36.00**	**33.28**	**33.18**

Table VIII: Average computational times of different algorithms for J30, J60 and J120 instances

Ins.	MFFE	Probability distribution				
		U1	U2	Exp	B1	B2
J30	5k	58.72	69.09	70.32	64.42	67.24
	25k	332.81	337.86	183.74	237.78	254.41
J60	5k	181.21	185.45	193.87	183.70	190.88
	25k	859.38	889.25	1103.19	511.12	927.85
J120	5k	1325.03	1325.05	1362.66	1190.88	1188.97
	25k	3563.33	4026.85	8947.86	8408.60	9110.82

Table IX: Variances of obtained solutions

Prob.	Variant	σ				
		U1	U2	Exp	B1	B2
J30	*var1*	3.76	9.19	19.11	3.83	10.07
	var2	3.75	9.23	**19.04**	**3.79**	**10.03**
	var3	**3.71**	**9.16**	19.44	3.87	10.12
J60	*var1*	4.20	**10.20**	20.81	**4.26**	10.94
	var2	4.22	10.22	**20.62**	4.27	**10.83**
	var3	**4.20**	10.25	20.65	4.28	11.03
J120	*var1*	4.16	10.27	**21.01**	4.25	11.11
	var2	4.13	10.34	21.64	4.24	**10.80**
	var3	**4.07**	**10.02**	21.09	**4.20**	11.08

Table X: Effect of N_S on performances of S-COA

N_S	J30			J60			J120		
	%Gap	Std	Time	%Gap	Std	Time	%Gap	Std	Time
10	7.94	18.40	38.94	11.23	20.64	98.98	39.58	21.09	662.66
50	11.79	18.72	39.55	28.70	20.65	193.87	53.61	21.63	726.64
100	16.66	19.44	70.32	32.22	20.94	216.68	57.38	22.69	819.00
1000	22.07	19.32	140.20	39.22	21.32	234.14	61.91	22.62	992.37

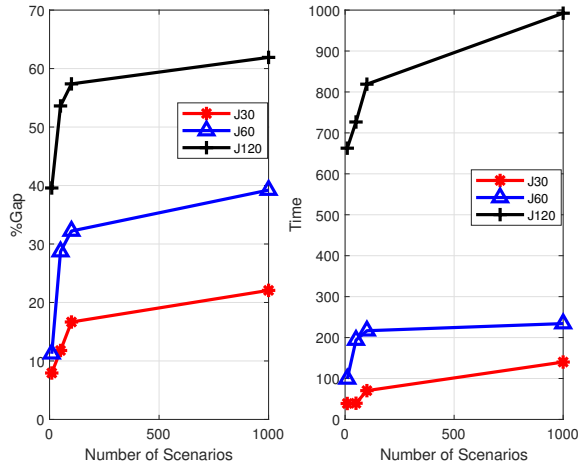

Fig. 5: Performance of S-COA for N_S

Table XI: Comparisons of different variants of proposed algorithm

Prob.	Variant	%Gap				
		U1	U2	Exp	B1	B2
J30	*var1*	**1.46**	8.69	17.90	**1.25**	7.91
	var2	2.52	11.26	21.49	2.32	10.41
	var3	1.56	**8.67**	**16.66**	1.29	**7.72**
J60	*var1*	12.85	19.53	29.63	12.65	**18.45**
	var2	14.37	23.62	36.77	14.19	22.73
	var3	**12.76**	**19.31**	**28.70**	**12.62**	18.54
J120	*var1*	34.80	37.42	41.14	34.71	35.84
	var2	38.31	45.73	58.98	38.35	44.66
	var3	**34.57**	**37.15**	**39.58**	**34.53**	**35.40**

Table XII: Computational times for different cases

Prob.	Variant	CPU Time (seconds)				
		U1	U2	Exp	B1	B2
J30	*var1*	126.47	145.62	175.78	121.05	189.61
	var2	68.54	72.90	77.75	70.34	99.22
	var3	**58.72**	**69.09**	**70.32**	**64.42**	**67.24**
J60	*var1*	386.22	512.26	491.71	572.90	551.64
	var2	269.27	273.42	271.13	289.08	285.06
	var3	**181.21**	**185.45**	**193.87**	**183.70**	**190.88**
J120	*var1*	2312.08	2340.11	2463.60	2229.42	2228.07
	var2	1472.18	1392.86	1453.09	612.01	1440.74
	var3	**1325.03**	**1325.05**	**1362.66**	**1190.88**	**1188.97**

E. Numbers of scenarios vs performances

In this section, we analyze the performances of the proposed algorithm, for the N_S (i.e., numbers of scenarios) considered in simulations. All the SRCPSP benchmarks are solved using S-COA with $N_S = 10$, 50, 100 and 1000, while MFFE is set to 5000 and the Exp distribution is used to generate the scenarios, as its σ is the maximum. Table X shows the gaps, Std and computational times (in seconds) for the J30, J60 and J120 benchmarks, and Fig. 5 the gaps and times with respect to the number of scenarios (N_S). Although both the gaps and times increase with increasing values of N_S, it is evident in Tables X and Tables V to VII, that S-COA still performs better than the state-of-the-art algorithms, even for a large number of scenarios.

F. Comparisons of different variants of S-COA

In this section, we analyze the performance of *var1*, *var2* and *var3* of the proposed S-COA for solving all the SRCPSPs with MFFEs of 5000 and 100, 50 and 10 scenarios for the J30, J60 and J120 problems, respectively. In Tables XI and XII, which present the results in terms of gaps and computational times for all the problems, respectively, it is clear that the quality of solutions obtained by *var1* and *var3* are almost similar (a statistical comparison is performed in subsection IV-H) and those by *var2* were worse while the computational times of *var1* for all problems in every PD are superior.

Fig 6 shows a normalized bar chart, in which an improvement in the average gaps of make-spans for *var3* over those for *var1*, and their corresponding CPU times for all benchmarks are depicted. It can be seen that the differences in make-spans are very small, while their CPU times are significantly different. Therefore, it can be concluded that *var3* performs better for all cases, based on the quality of solutions and computational time.

G. Effect on n

In this research, the random values of the durations are generated using different continuous PDs, which results in the values of d being continuous. However, we use a technique to round-off these values to tenths ($n = 10$) of decimal points, in order to deal with them in a more sophisticated manner in S-COA (see subsectionIII-B for more details), as discussed in this subsection.

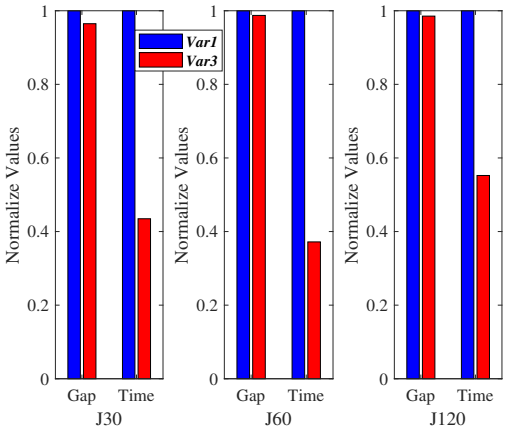

Fig. 6: Comparisons of average improvement in gaps of make-spans and corresponding CPU times

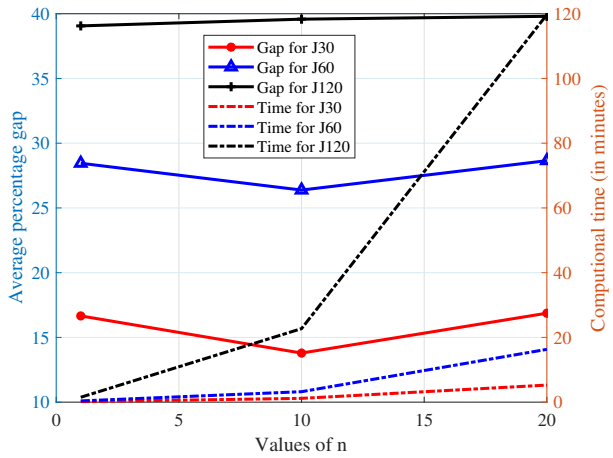

Fig. 7: Average gaps and computational times for different problems

All the SRCPSP benchmarks are solved using *var3* of the proposed algorithm and considering $n = 1$, $n = 10$ and $n = 20$. For this experiment, we set MFFEs to 5k, with an exponential distribution used to generate the scenarios' durations. In Fig. 7, the results show that computational time significantly increases when n increases but also that the average gaps do not change markedly. The minimum was found when $n = 10$, which means that the proposed algorithm is not sensitive (in terms of solution quality) to n, although it does perform better when $n = 10$.

H. Statistical test

In this section, the well-known Wilcoxon signed rank test is performed to determine the significant differences among the three variants of the proposed algorithm, by using them to solve the J120 benchmark set (600 problems) with five different PDs, U1, U2, Exp, B1 and B2.

For this test, a null hypothesis (H_0) is assumed, as:

$$H_0 : \ \alpha_1 = \alpha_2 = 0$$

where α_1 and α_2 are the median values of the two data sets obtained by two different variants of the algorithm. The H_0 is tested by considering a 5% significance level with the assumption that these median values are the same of the given data sets, where the value $H_0 = 1$ indicates that the null hypothesis is rejected, while $H_0 = 0$ indicates the alternative hypothesis that there is at least one of the values that is not zero.

Table XIII shows the H_0 values for different paired comparisons with various PDs. The experiments were carried out based on two parameters: (i) the average percentage of gaps; and (ii) the average Std of the FVs in the final generation, after applying the possible N_S scenarios. The last column shows test of the two variants for each experiment, based on the positive ranks of their gap percentages, Stds and CPU times. In the first experiment, comparing *var1* and *var2*, it is clear that *var1* is statistically better than *var2*, because it considers N_S scenarios from the first generation of evolution and fine-tunes the solutions after considering any possible uncertainty, while *var2* does not consider uncertainty until the last generation. However, in the next two experiments, it is clear that there are no significant differences in the quality of solutions obtained by *var1* and *var3*, and *var2* and *var3*, but *var3* is statistically better than *var1* and *var2* in terms of the gap percentage, Std and CPU time.

Table XIII: Statistical test results (H_0 values) by different variants of proposed algorithm for J30 benchmark set

Algorithms	Based on %Dev					Based on Std					Better
	U1	U2	Exp	B1	B2	U1	U2	Exp	B1	B2	
var1 vs. var2	1	1	1	1	1	0	0	1	0	0	*var1*
var1 vs. var3	0	0	0	0	0	0	0	0	0	0	*var3*
var2 vs. var3	0	0	0	0	0	0	0	0	0	1	*var3*

I. Comparisons with Monte-Carlo Simulation

In this section, to compare our results, we simulate the first test problem from each of the J11, J30, J60 and J120 instances using the well-known Monte Carlo (MC) simulation [29] and run it 1000 times, while considering a random scenario selected from the 1000 scenarios. Alternatively, in each run, the problem is solved deterministically while considering a random duration of each activity generated using a beta distribution (i.e., B1). The results from the MC simulations after 1000 runs, are compared with those obtained by the three variants of the scenario-based proposed algorithm, i.e., *var1*, *var2* and *var3*, from the final generation after applying 1000 scenarios. Table XIV shows the minimum (Min.), mean, median, maximum (Max.) and Std results of the make-spans for 1000 scenarios, with the computational times of the overall times required by the different approaches to evaluate the 1000 scenarios. In Table XIV, it is clear that the MC's computational times are significantly higher than those of the variants, with that of *var3* being the minimum. In terms of the quality of solutions, all the approaches produce almost the same solutions, that are not statistically significant at a 95% confidence level, as shown in Table XV. However, for larger problems, such as the J60 and J120 ones, the MC approach provides significantly better solutions, because its algorithm evaluates 5000 fitness functions for each random scenario, while in the scenario-based ones, only one fitness function is evaluated for each scenario, because 1000 scenarios are considered in each generation. Although, if the scenario-based approach could evaluate more fitness functions, it may obtain better results, this would be computationally expensive. Therefore, it can be concluded that a scenario-based approach, considering scenarios to the best individual in each generation (*var3*), performs best in terms of both solution quality and computational time.

Table XIV: Comparison of the results obtained by MC simulations and scenario-based algorithms

Prob.	Alg.	Make-span					Time
		Min	Mean	Median	Max.	Std	(min)
J11	MC	19.20	21.57	21.50	24.60	0.87	59
	var1	19.10	21.57	21.50	25.00	**0.86**	0.32
	var2	19.20	21.58	21.50	**24.20**	0.88	0.30
	var3	**18.80**	**21.54**	21.50	24.30	0.87	**0.074**
J30	MC	**35.20**	**45.00**	44.90	56.20	**2.97**	295.68
	var1	35.30	45.02	45.00	**54.50**	3.02	1.15
	var2	36.40	45.27	**44.90**	56.80	3.19	0.88
	var3	37.50	45.28	45.10	56.40	3.09	**0.86**
J60	MC	**63.60**	**77.41**	**77.30**	96.90	**4.58**	946.38
	var1	64.60	78.02	77.70	94.70	4.60	4.65
	var2	65.80	78.19	77.90	**94.60**	4.61	6.93
	var3	66.80	79.17	78.80	94.80	4.70	**0.90**
J120	MC	**99.10**	**109.99**	**110.00**	**123.40**	**3.73**	2346.30
	var1	102.30	112.15	111.70	124.60	3.87	30.87
	var2	100.90	112.39	112.30	124.80	**3.73**	34.45
	var3	99.50	113.42	113.40	129.40	4.10	**4.68**

Table XV: Statistical comparison of results obtained by MC simulations and scenario-based algorithms

Prob.	MC vs.					
	var1		*var2*		*var3*	
	H	p	H	p	H	p
J11	0	0.58	0	0.35	0	0.76
J30	0	0.38	0	0.08	0	0.05
J60	1	0.00	1	0.00	1	0.00
J120	1	0.00	1	0.00	1	0.00

V. Conclusion and Recommendations for Future Work

In this research, an efficient scenario-based approach for solving SRCPSPs with uncertain activity' duration has been developed. In the process, N_S scenarios were generated using five different probability distributions, with their parameters based on their forecasted values and historical errors. A new COA with two MOEAs, MOGA and MODE, was used to solve these SRCPSPs. It included a new operator which selected an elite solution in the following ways. Initially, all the individuals in the population in the g^{th} generation were evaluated, based on their deterministic values, and the best one was selected based on the minimum make-span. Then, the expected make-spans of the best individuals in the g^{th} and $(g-1)^{th}$ generations were re-evaluated considering N_S scenarios and finally, an elite individual was selected from them, based on their expected values. Once a given number of generations was completed, the elite solution obtained was called a 'robust schedule', as it could handle a large range of possible scenarios for uncertain activity durations. Since considering N_S scenarios for every individual in every generation was computationally expensive, we considered only the best ones, which resulted in a significant reduction in computational time.

For the experimental study, we considered standard RCPSP benchmarks with up to 120 non-dummy activities from the PSPLIB and their uncertain parameters of activity durations from different studies in the literature. The results obtained using the proposed approach, when compared with those of state-of-the-art algorithms, revealed that the former was superior in terms of solution quality with superior computational efficiency. This is a significant finding for scenario based approaches, for solving uncertain problems.

Possible future work could consider multi-mode RCPSPs, with both uncertain activity durations and uncertain resources. Furthermore, while we used a certain PD to determine the parameters for uncertain activity durations, which may not always be true in real life, adopting random values for them could be an improvement to this approach.

VI. Acknowledgement

This research is supported by a Australian Research Council Discovery Project (DP190102637) awarded to R. Sarker, D. Essam and C. Coello Coello.

References

[1] S. Elsayed, R. Sarker, T. Ray, and C. C. Coello, "Consolidated optimization algorithm for resource-constrained project scheduling problems," *Information Sciences*, vol. 418-419, pp. 346 – 362, 2017.

[2] H. Li and N. K. Womer, "Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming," *European Journal of Operational Research*, vol. 246, no. 1, pp. 20 – 33, 2015.

[3] F. Ballestín, "When it is worthwhile to work with the stochastic rcpsp?" *Journal of Scheduling*, vol. 10, no. 3, pp. 153–166, Jun 2007.

[4] F. Ballestï¿œn and R. Leus, "Resource-constrained project scheduling for timely project completion with stochastic activity durations," *Production and Operations Management*, vol. 18, no. 4, pp. 459–474, 2009.

[5] M. Bruni, L. D. P. Pugliese, P. Beraldi, and F. Guerriero, "An adjustable robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations," *Omega*, vol. 71, no. Supplement C, pp. 66 – 84, 2017.

[6] B. Ashtiani, R. Leus, and M.-B. Aryanezhad, "New competitive results for the stochastic resource-constrained project scheduling problem: exploring the benefits ofÂ pre-processing," *Journal of Scheduling*, vol. 14, no. 2, pp. 157–171, Apr 2011.

[7] R. K. Chakrabortty, R. A. Sarker, and D. L. Essam, "Resource constrained project scheduling with uncertain activity durations," *Computers & Industrial Engineering*, vol. 112, no. Supplement C, pp. 537 – 550, 2017.

[8] M. Zaman, S. M. Elsayed, T. Ray, and R. A. Sarker, "Evolutionary algorithms for power generation planning with uncertain renewable energy," *Energy*, vol. 112, pp. pp. 408–419, 2016.

[9] W. Herroelen and R. Leus, "Project scheduling under uncertainty: Survey and research potentials," *European Journal of Operational Research*, vol. 165, no. 2, pp. 289 – 306, 2005, project Management and Scheduling.

[10] M.-Y. Cheng, D.-H. Tran, and Y.-W. Wu, "Using a fuzzy clustering chaotic-based differential evolution with serial method to solve resource-constrained project scheduling problems," *Automation in Construction*, vol. 37, pp. 88–97, 2014.

[11] S. Rostami, S. Creemers, and R. Leus, "New strategies for stochastic resource-constrained project scheduling," *Journal of Scheduling*, Jan 2017.

[12] C.-C. Tseng and P.-W. Ko, "Measuring schedule uncertainty for a stochastic resource-constrained project using scenario-based approach with utility-entropy decision model," *Journal of Industrial and Production Engineering*, vol. 33, no. 8, pp. 558–567, 2016.

[13] C. Fang, R. Kolisch, L. Wang, and C. Mu, "An estimation of distribution algorithm and new computational results for the stochastic resource-constrained project scheduling problem," *Flexible Services and Manufacturing Journal*, vol. 27, no. 4, pp. 585–605, Dec 2015.

[14] R. Kolisch and A. Sprecher, "Psplib - a project scheduling problem library," *European Journal of Operational Research*, vol. 96, no. 1, pp. 205–216, 1997, cited By 622.

[15] M. F. Zaman, S. Elsayed, T. Ray, and R. Sarker, "Scenario-based solution approach for uncertain resource constrained scheduling problems," in *IEEE Congress on Evolutionary Computation,*, 2018.

[16] S. Creemers, "Minimizing the expected makespan of a project with stochastic activity durations under resource constraints," *Journal of Scheduling*, vol. 18, no. 3, pp. 263–273, Jun 2015.

[17] J. Alcaraz and C. Maroto, "A robust genetic algorithm for resource allocation in project scheduling," *Annals of Operations Research*, vol. 102, no. 1-4, pp. 83–109, 2001.

[18] D. Debels and M. Vanhoucke, "A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem," *Operations Research*, vol. 55, no. 3, pp. 457–469, 2007.

[19] V. Valls, F. BallestÃn, and S. Quintanilla, "A hybrid genetic algorithm for the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 185, no. 2, pp. 495–508, 2008.

[20] I. Ali, S. Elsayed, T. Ray, and R. Sarker, "A differential evolution algorithm for solving resource constrained project scheduling problems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9592, pp. 209–220, 2016.

[21] Q. Jia and Y. Seo, "An improved particle swarm optimization for the resource-constrained project scheduling problem," *International Journal of Advanced Manufacturing Technology*, vol. 67, no. 9-12, pp. 2627–2638, 2013.

[22] F. Ballestï¿œn and R. Leus, "Resource constrained project scheduling for timely project completion with stochastic activity durations," *Production and Operations Management*, vol. 18, no. 4, pp. 459–474, 2009.

[23] B. Ashtiani, R. Leus, and M.-B. Aryanezhad, "New competitive results for the stochastic resource-constrained project scheduling problem: exploring the benefits of pre-processing," *Journal of Scheduling*, vol. 14, no. 2, pp. 157–171, Apr 2011.

[24] A. A. Fernandez, R. L. Armacost, and J. J. Pet-Edwards, "Understanding simulation solutions to resource constrained project scheduling problems with stochastic task durations," *Engineering Management Journal*, vol. 10, no. 4, pp. 5–13, 1998.

[25] M.-S. Chang, Y.-L. Tseng, and J.-W. Chen, "A scenario planning approach for the flood emergency logistics preparation problem under uncertainty," *Transportation Research Part E: Logistics and Transportation Review*, vol. 43, no. 6, pp. 737 – 754, 2007, challenges of Emergency Logistics Management.

[26] M. F. Zaman, S. M. Elsayed, T. Ray, and R. A. Sarker, "Evolutionary algorithms for dynamic economic dispatch problems," *IEEE Transactions on Power Systems,*, vol. 31, pp. 1486–1495, 2016.

[27] Y.-W. Tsai and D. D. Gemmill, "Using tabu search to schedule activities of stochastic resource-constrained projects," *European Journal of Operational Research*, vol. 111, no. 1, pp. 129 – 141, 1998.

[28] Z. Chen, E. Demeulemeester, S. Bai, and Y. Guo, "Efficient priority rules for the stochastic resource-constrained project scheduling problem," *European Journal of Operational Research*, 2018.

[29] Y. H. Kwak and L. Ingall, "Exploring monte carlo simulation applications for project management," *Risk Management*, vol. 9, no. 1, pp. 44–57, Feb 2007.