International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V72I12P114

Volume 72 Issue 12, 151-159, December 2024
© 2024 Seventh Sense Research Group®

Original Article

The Scheduling Problem of Renewable Resources Adapted by a Competitive Differential Algorithm

Amol Chaudhary[1], Sachin Meshram[2]

[1,2]Mechanical Engineering Department, G H Raisoni University, Amravati, India.

[1]Corresponding Author : amolchaudhary50@gmail.com

Received: 20 June 2024 Revised: 20 October 2024 Accepted: 02 December 2024 Published: 25 December 2024

Abstract *- Valuable asset bound project Planning with Inexhaustible assets is a point that has been quite examined. Basically, you have a list of things you need to do in a project, and you have a limited amount of stuff you can use to do those things. There is only one way to do each task, and the resources can be used repeatedly. To address this problem, we used Differential Evolution (DE) to discover first-class work scheduling in an undertaking so that you can reduce the entire mission's final touch time (referred to as the makespan). The number of sources to be had and the duties that may be completed earlier than or after one another (referred to as precedence regulations) should be considered. We used a sequential era method to assemble the challenge schedule and a prioritized fee-driven method to describe it if you want to streamline the procedure. Subsequent, we contrasted our DE set of rules with some different algorithms that had already been created by using other academics. Using Patterson's check bed, we assessed each one and discovered that the DE algorithm finished pretty well, supplying attainable solutions for most problems. This is a remarkable new approach to the problem of RCPSP involving scarce and renewable resources.*

Keywords *- DE algorithm, RCPSP, Time period optimization, Branch & Bound Approach, Makespan.*

1. Introduction

Task managers confront a challenging project when faced with the mission scheduling challenge with limited resources (RCPSP), which requires careful planning and proper sequencing while also dealing with resources like time, money, and labor. Schedule the use of all renewable and limited resources. There are three basic approaches to addressing this issue: precise techniques, heuristics, and a near-optimal course of action. Definite techniques strive to discover the optimal answer, but they may take a long time and may not always be effective for large projects. Heuristics are more like rules of thumb that help you develop a decent answer fast, but they may not necessarily provide the greatest solution. Meta-heuristic approaches are similar to intelligent strategies that try to find the best answer by investigating numerous choices, albeit they may take some time. Researchers have written countless publications on such responses throughout the years, recommending ways to boost their potential. The study references many well-known experts on the issue, including Hartmann and Kolisch [1], Kolisch and Padman [2], and Kolisch and Hartmann [3]. RCPSP is classified as an NP-hard optimization problem, as previously stated. While bigger operations are concerned, using real algorithms may result in excessively lengthy execution durations. Metaheuristic methodologies are used in various research projects to solve this issue. Here is a quick review of some of those tactics. Among these tactics is the genetic set of rules (GA), which is entirely based on global evolution and herbal selection concepts. It was successfully used to handle a variety of project scheduling challenges [4]-[11]. It compares issues that must be addressed to a network of living objects found in the environment, with the answers preserved in something known as a "chromosome."Rather than depending on sheer force to find the best solution, the method uses specialized operators to improve the fitness of the chromosomes. By simulating natural selection, GA produces a more efficient and adaptive outcome.

Zamani (2013) [12] developed a novel genetic algorithm that stands out for using a magnet-based crossover operator. This operator may keep up to two connected segments from the recipient and one from the donor in their genetic composition. This guarantees that the recipient's genes correspond with the donor's genetic composition. This crossover operator retains up to three interlocked segments rather than merely distinguishing it from the usual-factor crossover. Only two connected segments are kept inside the -factor crossover and must come from the same figure.

GA handles optimization problems using simulated annealing, a probabilistic approach. This approach incorporates "annealing," which involves gradually reducing the temperature so the algorithm can discover a high-quality solution. While it has reached its lowest temperature. GA may address task scheduling challenges, as shown in studies by Bouleimen and Lecocq [13] and Boctor [14]. It does this by avoiding being trapped in neighboring peaks. ACO is used in this method. This strategy is comparable to how ants look for food. It works by regularly altering pheromone pathways among a group of simulated ants, enabling them to choose the optimum solution while adhering to the rules. Merkle et al. [15] and Lo et al. [16] have described this method more specifically.

Zhang et al. (2006) [17] demonstrate how PSO has been used for scheduling challenging scenarios. Debris in PSO moves about the answer area, each representing a possible solution. The PSO and DE algorithms and the artificial Bee Colony (ABC) approach proportionally manage factors such as colony length and cycle amount.

Jia [18] and search engine optimization (2013) improved the efficiency of the helpful resource-restricted mission scheduling hassle (RCPSP) by using the facility layout hassle (FLP) technique. They developed new ways using the Permutation-primarily based synthetic Bee Colony (PABC) principles. The first method is to immediately use the FLP concept to plot activities within time and aid limitations with no conflicts or collection violations occurring. While the second strategy is similar to the first, it tackles large instances of the problem categorized as NP-hard using the Permutation Illustration Scheme from the PABC set of rules.

They eventually added a new layer of complexity by using Variable Neighborhood Search (VNS) to find virtually flawless solutions. VNS is related to a strategy that systematically adjusts the environment and employs basic local search techniques for greater solutions. It has already been used to solve scheduling challenges, as shown by Fleszar and Hindi's 2004 study. This paper studies how to calculate RCPSP and how to use it in the Differential Evolution (DE) meta-heuristic technique. In 1997, Storn and Fee [19] created DE, a powerful approach that combines classic crossover, mutation, and reputation operators with basic mathematical operators. DE's core system begins with developing trial parameter vectors, which are subsequently formed by crossover and mutation. Choice determines which vectors will be surpassed down to the next period.

The use of DE to address scheduling challenges in projects has been investigated. For example, Damak et al.;2009 [20] employed DE to solve resource-constrained challenge scheduling problems (MRCPSP) in a variety of modalities. This approach represents solutions as a vector of coordinates and a vector of mode assignments. The choice operator penalizes unrealistic solutions based on the purpose characteristic's values. Preferred scenarios were used to test the algorithm's performance, and the results were compared to those of particle swarm optimization by Jarboui et al. (2008) [21] and simulated annealing by Bouleimen and Lecocq [13]. A differential evolution (DE) technique change was used by Rahimi et al. (2013) [22] to resolve the mode identity constraints venture scheduling problem (MIRCPSP). The approach was coupled with a module for nearby search and gaining knowledge to improve the current DE's efficacy. The objective function and computing instances of the solutions to several take a look at issues have been compared statistically to evaluate the performance of the DE.

2. Resource-Constrained Project Scheduling Problem (RCPSP)

The issue with project planning is sorting out what amount of time it will require to do an undertaking's exercises to accomplish a specific objective. In primary research on project scheduling, it was assumed that a project's activities are only described by the time it takes to carry them out. After that, strategies like the basic way strategy (CPM) and the program assessment and survey method (Energetic) are suggested by considering the most important connections between tasks. Constraints will be applied, and their effects will be evaluated because assuming that precedence relationships are independent appears unreasonable. The Resource-Constrained Project Scheduling Problem, or RCPSP, is considered a typical project scheduling issue. This problem considers a project with J activities labeled j=1,..., J. They are one of the major limitations of project scheduling. In addition, the duration of an activity j, or processing time, is represented by dj, indicating that the activity must be finished immediately after beginning. Technological requirements are generally placed in order of precedence among the activities. These connections are shown by sets of prompt ancestors Pj, demonstrating that an action j cannot be begun before the fruition of every one of its ancestors (iεPj). Furthermore, these connections can be addressed as an organization. A specific measure of assets is expected for every movement to be performed. Attributable to a full limit accessible in each period, the assets are perceived as sustainable. Altogether, we have K sorts of inexhaustible assets named k=1,…, K. It is assumed that a constant amount of Rk is available prior to the beginning of each period for each resource k. It is necessary to have rjk units of resource k in each period where the processing is carried out to carry out activity j. Additionally, two additional activities, j=0 and j=J+1, signify the project's beginning and end, respectively, are considered. Since both are fictitious activities, there is no processing time or use of resources. The problem's information is taken to be definite and deterministic, and the parameters are taken to be integer-valued and non-negative. In this issue, the point is to carve out the beginning opportunity (Sj) for the exercises j=0,1,…, J+1, such that the consummation season of the venture is limited Hartmann, 2002[5]. As previously mentioned, this is the fundamental form of the resource-constrained project scheduling problem. Nonetheless, since the circumstances are totally divergent practically, changes have been made to the essential suspicions by the analysts over the long run. In addition, the RCPSP problem is one of the strongly NP-hard problems, as demonstrated by Blazewicz et al. [23]. As a result, various solutions are utilized based on the changes in the fundamental assumptions.

3. Problem Definition

Experience demonstrates that resources are incredibly crucial for every project. This phase will deal with the fundamental resource-restrained undertaking scheduling hassle, typically called the RCPSP. It is a common issue involving splitting time into specific periods and breaking tasks into various activities. We will have a collection of n activities, each with a duration di, where i ranges from 1 to m. Certain activities cannot begin until others have been completed, necessitating us to consider this factor. There are several periods of time within the planning span, and the number of activities needed for each job is limited to a certain amount. To make things more organized, we will label all the activities from A0 to An+1, where A0 is the first thing we need to do (the source) and Am+1 is the last thing we need to do (the sink). Without a source or sink, a dummy task will be included with zero duration and no resource needs. However, every task requires resources to be completed. The total number of resources is R, and we can only have a maximum of Rj units of resource j at any given period. The resources are renewable, so we can use them as much as we want, but we can only use them when they are available. The RCPSP aims to parent out a way to do all the duties using the least amount of resources. To do this, we must determine when each task starts and finishes, considering all the resources we need and the order of things.

The RCPSP is an integer programming problem that can be represented by xkt as a variable set to 1 if task Ak is scheduled to finish at the end of period t and 0 if not. For every task, Ak is assigned the Earliest Finish Time (EFT), EFk, and the latest finish time (LST), LFk, which is allocated using the Kelley and Walker (1959) technique. LFm+1 is assigned a value of H, which will never exceed the total duration of all activities.

$$\text{Minimize } \sum_{t=EF\ m+1}^{t=LF\ m+1} tXm+1,t \qquad (1)$$

$$\text{Subject to : } \sum_{t=EFk}^{LFj} Xkt = 1 \text{ for k=0,…..,n+1} \qquad (2)$$

$$\sum_{t=EFi}^{LFi} tXit \le \sum_{t=EFk}^{LFk} tXkt - dk \qquad (3)$$

for all (A_i, A_j) ε P

$$\sum_{q=\max\{t,EFk\}}^{\min\{t+dk-1,LFk\}} rjkXjq \le R_j \text{ for k=1,..R} \qquad (4)$$

$$x_{kt} \ \varepsilon \ \{0,1\} \text{ for i=0,…..,m+1; t= } EF_k,\dots,LF_i \qquad (5)$$

(2) Constraints ensure that each task is carried out only once. P is the assortment of all sets of exercises (Ai, Ak) with computer-based intelligence going before Ak, addressed by disparities (3) to show the prioritized imperatives. The use of binary decision variables is specified by restriction (5), and restriction (4) ensures that the utilization of each resource does not continue to exceed the quantity that is currently available. An ideal schedule showing the times at which each task can be completed can be obtained by solving problems (1) to (5).

4. Implementation

Our work's Genetic Algorithm is a combination of proposed and proven components. Literature-collected proven components are adapted for use by incorporating variations. Controlled random numbers have been extensively used, as in most genetic algorithms.

From a convenience standpoint, we use software and platforms that are widely available for implementation. This also puts us on par with the majority of other research, making it easier to make straightforward comparisons.

4.1. Stage Depiction (Equipment, Programming, and so forth)

The algorithm is implemented in Structured C and compiled using Borland® C++. We took advantage of a compiler feature unique to (Kernighan/Ritchie) C by doing so. For instance, we consolidated elements of C++ for record perusing and composing. Under Microsoft Windows XP, the program is run on a 2GHz Intel Pentium4 computer with 2GB of RAM. Contingent upon informational collection and boundaries chosen for testing and observing, the run time for a complete informational collection went from under seven minutes (averaging 875 milliseconds for each occurrence) to simply over 24 hours (averaging three minutes for every occasion).

4.2. Input Information- The Test Data-Set

4.2.1. The Input Data-set

To evaluate scheduling strategies for the RCPSP, we put our algorithm through its paces on internationally recognized standard benchmark instances provided by Kolisch in 2000 [1]. It is known as PSPLIB, and the scientific literature acknowledges its usefulness. We used the PSPLIB's standard SMFF (Single Mode, Full Factorial) set as test cases. The Project lengths are indicated by the labels J30, J60, J90, and J120 on these. We used J30 and J60 as our test data set because they each have (48 X 10 =) 480 project instances.

Every one of the sets manages four compelled inexhaustible assets. One mode of execution governs each task. The interconnectedness of the task dependencies, the number of resource types, and the quantity of available resources are roughly equivalent to the three parameters on which this set is dependent. The PSPLIB also provides SMCP, MMCP, and MMFF sets as test problems. The library moderators regularly compile a performance comparison between the algorithms of various researchers who utilize this data set. We use Kolish and Hartmann's (2006) [3] refreshed correlation for benchmarking our exploratory outcomes. In this case, the paper's authors have invited "future studies" to use the compiled results as benchmarks. Hartmann and Kolisch [1] and Kolisch and Padman [2] made previous comparisons in the literature in 2000 and 1996. The most recent, up-to-date list of the best results on each instance set by various researchers can be found in the same library.

4.2.2. Other Test Data-set

SMFF informational collection of PSPLIB is the most generally involved test information for the RCPSP. This library contains additional benchmarking-useful standard data sets and those available from other libraries and authors. A short portrayal of these is given here.

a) PAT: James Patterson presented this simple arrangement of occurrences in his correlation of careful arrangement techniques for asset-obligated project planning. There are 110 project scheduling issues in the Patterson set (PAT) whose tasks have a single execution mode and multiple resources. The optimal resource-constrained solution is frequently identical to the optimal resource-unconstrained solution because the resource constraints are not particularly strict.

b) SMCP: The Single Mode Ceteris Paribus set is similar to the Patterson set but has more resource restrictions and ten to forty tasks. There are 200 problems in the set, ranging from one to four types of renewable resources. There is only one execution mode for each task.

c) MMFF: The Multi-Mode Full Factorial set includes four different resource types, two of which are renewable and two of which are not. Only about 85% of the cases in this set are known to have solutions that could be implemented. The addition of non-renewable resources raises the possibility of creating unsolvable issues.

d) BMRX: At the beginning of 1995, Barry Fox and Mark Ringer proposed the Bench MaRX problem. There are twelve parts to this one problem. Each component tests various aspects of a solution strategy by adding additional constraints or problem modifications. The first four sections use pretty standard language. From there, it gets harder. There are 575 tasks, three types of labor resources, and fourteen location-based resources in the problem. It includes numerous temporal restrictions in addition to resource and location constraints, such as three shifts per day, resources restricted to specific shifts, and task start and finishes required within a shift or permitted to cross shifts. The final of the twelve sections has multiple goals. By fluctuating asset accessibility and work orders after a timetable is not entirely settled, the issue likewise tests the capacity of arrangement techniques to adjust to dynamic changes.

e) Boctor Sets: Boctor has given a bunch of test information, which 1s named as boctor 50mm boctor 100mm.

f) Alvarez-Tamarit sets: The authors created prob 103, prob 27, and prob 51, three test data sets. As previously stated, we have experimented with the Kolisch [1] SMFF set. We refer to the PSPLIB for additional information on the instance generation mechanism, the instance generator (Pro Gen), solution sets, and other topics. We portray our test information (SMFF) arrangement as given in the PSPLIB in Figure. The library has converted it from Pro Gen format to Patterson format. Let us denote

j=1,…..;J	: jobs
r=1…..,R	: resource types
S(j)	: number of immediate successor-jobs of job j
S(j,s)	: s-th immediate successor-job of job j
D(j)	: non-preemptable duration of job j
K(r)	: resource availability of resource type r within each period
K(j,r)	: resource usage of job j w.r.t. resource type r

Keep in mind that the instance size (j) for a project with 30 activities is 32. The initial and conclusion points are dummy tasks that make up the two additional tasks. (For the purposes of this discussion, the term "tasks" refers to a toy or a project size that includes the dummy activities.)

An example of a test project with 32 tasks (or 30 activities) is provided based on the above format definition. Even though we used the SMFF data set, we were able to test our algorithm's adaptability to other datasets by making a small change to our program.

The format is :

J	R							
K(1)	K(2)	..	..	K(R)				
d(1)	k(1,1)	..	k(1,R)	S(1)	S(1,1)	...	..	S(a,S(1))
..								
..								
d(J)	k(J,1)	..	k(J,R)	S(J)	0			

Table 1. Instance example of the J30 test data-set; File name : J301_1.rcp

32	4							
12	13	4	12					
0	0	0	0	0	3	2	3	4
8	4	0	0	0	3	6	11	15
4	10	0	0	0	3	7	8	13
6	0	0	0	3	3	5	9	10
3	3	0	0	0	1	20		
8	0	0	0	8	1	30		
5	4	0	0	0	1	27		
9	0	1	0	0	3	12	19	27
2	6	0	0	0	1	14		
7	0	0	0	1	2	16	25	
9	0	5	0	0	2	20	26	
2	0	7	0	0	1	14		
6	4	0	0	0	2	17	18	
3	0	8	0	0	1	17		
9	3	0	0	0	1	25		
10	0	0	0	5	2	21	22	
6	0	0	0	8	1	22		
5	0	0	0	7	2	20	22	
3	0	1	0	0	2	24	29	
7	0	10	0	0	2	23	25	
2	0	0	0	6	1	28		
7	2	0	0	0	1	23		
2	3	0	0	0	1	24		
3	0	9	0	0	1	30		
3	4	0	0	0	1	30		
7	0	0	4	0	1	31		
8	0	0	0	7	1	28		
3	0	8	0	0	1	31		
7	0	7	0	0	1	32		
2	0	7	0	0	1	32		
2	0	0	2	0	1	32		
0	0	0	0	0	0			

There are four parts to each data set:

- Tasks (or activities, or nodes, depending on which term the authors prefer),
- Assets (we zeroed in on the amount compelled sustainable assets),
- Precedence Constraints (which predecessor or successor is shown)
- Time taken to finish the task

The position format of the preceding four components is the primary difference between the input and output data sets. Whenever this is broken down and grasped, and imperative changes are made in the 'input data' capability, the rest of our program continues as before. However, the Single Mode aspect was our primary focus at all times.

5. Differential Evolution (DE)

DE is based on populations and randomness, which takes inspiration from Evolutionary Algorithms (EAs) in general and, more specifically, the Genetic Algorithm (GA). It tries to find a solution close to optimal or satisfactory by combining existing ones using techniques like mutation, crossover, and acceptance.

In differential advancement, people are signified by D-layered vectors known as x_i, with D addressing the number of genuine boundaries and NP as the populace size. Beginning with a group of beings with randomly generated values for their components (genes), evolution begins. Initially, the mutation process takes place by randomly selecting three individuals and blending their gene values in various ways. More specifically, the differences in gene values between the other two individuals and the initial person are combined. This procedure can be summed up as follows:

$$DE = (x_{i1} - x_{i2}) + x_{i1} + (x_{i2} - x_{i3}) + x_{i2} + (x_{i3} - x_{i1}) + x_{i3}$$

$$Vi= x_{i1} + F \ldots\ldots (x_{i2}- x_{i3} \qquad (6)$$

i_1, i_2, i_3, and ε are distinct integers between 1 and NP (the population size) are distinct integers that do not include the vector's index i. The difference between two vectors is affected by the constant F greater than zero. A crossover between the mutated individual (vi) and a target individual from the current population (xi), known as the trial individual, produces a trial individual. During the crossover operation, it is necessary to abide by these regulations.

$$y_i(j) = \{v_i(j), \text{ if } R_j \leq cr \text{ or } j = j_{random} \quad (7)$$

x_i (k), otherwise

The crossover rate constant, cr ε is defined as (0,1), and each R_j is randomly selected with values between 0 and 1. In this simulation, there are D possible values for randomly chosen j, from the set {1, 2, ..., D}, for each individual. Following mutation and crossover, the trial individual is then subject to acceptance. The trial individual's fitness is evaluated against the target individual's, and the one with the higher fitness level is chosen to advance to the next generation. The trial person's fitness is then compared to that of the fitter person, and if it is higher, the trial person is chosen as the new target person for the next generation.

$$x_i = \{y_i, \text{ if } f(y_i) \leq f(x_i) \quad (8)$$

x_i, otherwise

A fresh group of people is formed by following the same steps with each person from the initial group, and this cycle continues until a specific requirement is fulfilled. The solution to the problem is determined by selecting the top individual from the previous generation.

5.1. Representation of a Solution

A vector (I) of size m with one element for each task solves this issue. The kth element represents the priority of task k in the solution (a priority list). The kth element pv_k^I ε {1,2,…..,n} , which can have values from 1 to m:

$$I = (pv_1^I, pv_2^I, \ldots pv_m^I) \quad (9)$$

The Serial Schedule Generation Scheme (SSGS) creates a schedule for a single person. Essentially, SSGS focuses on the makespan criterion, a standard measure of performance that improves as task completion times decrease. Using the serial SGS guarantees that no best schedules will be overlooked, allowing us to utilize it for this task confidently. With a specific individual I, we can determine their respective schedule by adhering to these steps:

- Check if there is only one task for I. If so, then the schedule is just that one task.
- Otherwise, we make a list of all the activities in increasing order of their completion times.
- Then, we go through the list and add each task to the schedule one at a time, in the order they appear in the list. The first task is added to the schedule immediately, while the last one is added at the end.
- Once we have added all the activities, the schedule is complete.
  1) Let P= 1.
  2) Set the dummy start task 1 at time 0.
  3) P= P+ 1.
  4) Select the task k with the priority value pvkI = P.
  5) Compute the earliest precedence and resource feasible start time of task k.
  6) If the dummy end task m is on the schedule, halt. If not, proceed to step 3.

Table 2. Time needed for operation

Task	**Task**	**Duration (Shifts)**	**Resource**		
			Type A	**Type B**	**Type C**
1-2	Gearbox taking out	2	1	1	3
1-3	Bottom frame Removing	2	1	1	5
2-4	Bottom frame fabrication	2	1	1	4
3-5	Gearbox fitting	2	1	1	4
4-6	Gearbox alignment	2	1	1	5
5-7	Motor fitting and alignment	2	1	1	4
6-8	barrel lifting	2	2	1	4
7-9	Releasing J1 pin	2	1	1	4
8-10	Releasing J2 pin	2	1	1	4
9-11	Barrel taking out	1	2	1	4
11-12	Barrel lifting (other side)	1	1	1	5
12-13	Making J3 pin free	2	1	1	4
13-14	Making J4 pin free	1	1	1	4
14-15	Barrel taking Opposite side	1	1	0	4
15-16	Bottom frame adjusting	1	0	0	1
10-16	addressing oil leaks in packages and valve maintenance	2	2	1	3

Table 3. Resource accessibility

Type	Resource	Max. Resource Accessibility
A	Fitter	2
B	Welder	2
C	Helper	8

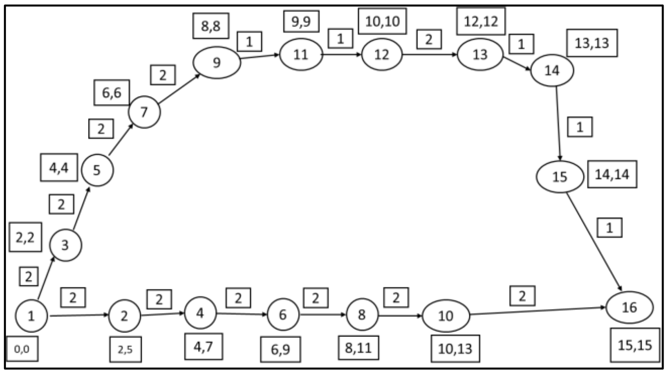
Fig. 1 Network layout for various maintenance tasks

5.2. Original Population

We begin with an empty m-element vector and repeat the following steps to create a priority value list based on the precedence constraints for each individual I in the initial population:

1. Randomly choose an eligible task k.
2. Choose a random index m from the set {1, 2, ..., m} (representing the order numbers from 1 to m).
3. Assign the priority value of task k (pv_K^I) to index m.

This cycle is rehashed for a pre-indicated number of arrangements equivalent to the size of the pop-size.

5.3. Scheduling Problem

In particular, this research focuses on planning when maintenance tasks for the Wagon tippler will take place. The time needed for each maintenance task is determined by the length of regular work shifts, which are eight hours in duration.

5.4. Genetic Ge-combination

In the following manner to members of the previous generation:

5.4.1. Mutation Operators

Suppose we have a present population containing individuals labeled I_{i1}, I_{i2}, and I_{i3}, ranging from 1 to NP. We select a trio of these persons labeled as I_1, I_2, and I_3. Next, we aim to generate a fresh mutant individual by executing a mutation operation on their priority lists. We achieve this by randomly selecting three individuals and utilizing a factor F_{pv}. Using the equation, we apply the mutation operation to their priority lists using the scale factor F_{pv}.:

$$v_{ni}=pv_{i1}+F_{pv}\,(pv_{i2}-pv_{i3})$$

Where individual i's priority value list is pv_x and v_{ni} is the mutant. A new mutant with a new priority list v_{ni} is created as a result.

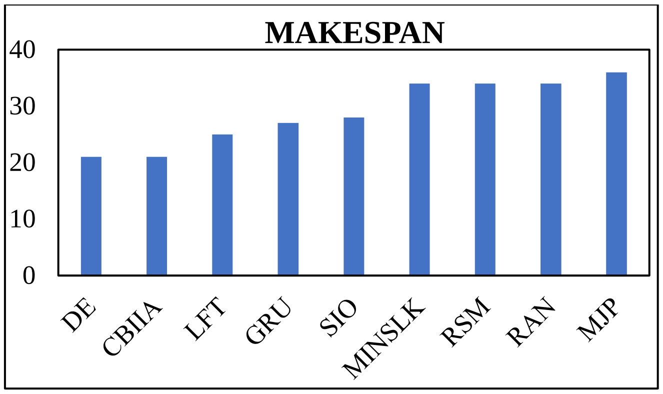

Fig. 2 Suggested DE in comparison

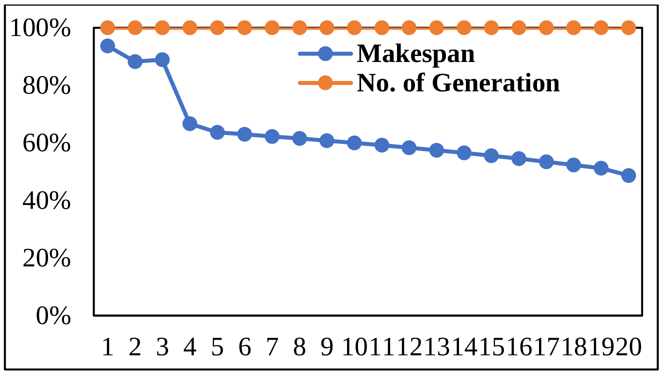

Fig. 3 Makespan value with a number of generations

5.4.2. Crossover Operators

With a crossover rate of cr_{pv}, the crossover operator combines bits from two individuals—one from the current population and the other from a mutant—to create a new member. In order to accomplish this, it selects a random number R_j for each element k in the priority list from a uniform distribution between 0 and 1.

If R_j is equal to or lower than cr_{pv}, the trial individual copies the element from the mutant individual; if not, it copies the target individual. Following the crossover operator, the new individual's structure may not align with the intended solution representation, as elements could have unexpected values. In order to correct this issue, the test subject is reverted back to the initial form using the following process:

1. Set a counter l to 1.
2. Give the dummy start task a priority value of 1.
3. Increment l.
4. Find the set of eligible activities (EK) by looking at which activities have already had their predecessor's plan.
5. Choose the eligible task k with the lowest precedence:
1. $Pv_k^I = \min \{ pv_k^I I \varepsilon\ EK\}$.
2. Set the pv_k^I priority value to be the value of l priority.
6. If the dummy end task m is part of the eligible activities, stop; otherwise, go to step 3.

Table 4. Schedule obtained through differential evolution

Activities Considered	1-3	1-2	3-5	2-4	5-7	5-7	4-6	11-12	6-8	12-13	8-10	13-14	14-15	10-16	15-16
Makespan	2	2	4	4	6	8	10	12	14	16	16	17	18	20	21
Total Make Span = 21															

This conversion process ensures that the trial individual's priority values align with the original representation, even if the crossover operator has messed them up.

Acceptance operator: When the transformation and hybrid activities have been finished, we consider the preliminary person's goal capability esteem in contrast to the objective person's. The trial person will be chosen for the next generation if the evaluation result is the same or lower than the target person's. The intended recipient moves on to the next generation if it is higher.

6. Execution and Balancing

The widely recognized Patterson experimental setup evaluated the suggested algorithm against different techniques in the published studies (Wu et al., 2011). This test environment consists of 16 tasks, with dummy activities at the beginning and end. Three different resources are needed in specific quantities for every task. The data required to address the issue is presented in Figure 1. Per the experiment setup, there are 16 activities, all with predetermined unit demands for three resources. The specifics of the issue are outlined in Figure 1. Implementing the DE algorithm on Patterson's test bed revealed a total makespan 21. The resulting timetable is shown in Table 3. Figure 2 compares the DE algorithm and other methods; Wu et al., 2011. The DE algorithm performs better than others, as illustrated in the diagram. Both the DE and CBIIA algorithms have a makespan of 21. However, the DE algorithm can reach this solution in only 0.1 seconds compared to the 0.51185 seconds required by the CBIIA algorithm. Hence, the DE algorithm is the top choice regarding the time needed to achieve a solution close to optimal. Aside from the makespan, the number of iterations required to achieve the final solution is also crucial when comparing the two algorithms. The DE algorithm achieves a great solution in just 20 iterations, whereas the CBIIA algorithm takes 52 iterations to reach a solution. This pattern demonstrates that exploring the solution space for larger problem sizes will take longer iterations. Considering both makespan and the number of iterations, the DE algorithm with fewer iterations is preferable. Figure 3 displays the convergence pattern of the DE algorithm.

7. Conclusion

In this study, we presented the widely recognized challenge of RCPSP for its high computational complexity. The significance of reducing the makespan is clear in today's competitive business landscape. Because the problem is NP-hard, near-optimal methods are needed to solve it efficiently. Consequently, a productive evolutionary technique is known as DE (Differential Evolution). Next, we evaluated the outcomes by comparing them with various algorithms already documented in research. The findings indicated that DE performed better than the approaches it was compared in terms of both solution quality and computational efficiency, demonstrating its potential as a robust method for addressing the RCPSP challenge.

References

[1] Sönke Hartmann, and Rainer Kolisch, "Experimental Evaluation of State-of-the-Art Heuristics for the Resource-Constrained Project Scheduling Problem," *European Journal of Operational Research*, vol. 127, no. 2, pp. 394-407, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[2] Rainer Kolisch, and Rema Padman, "An Integrated Survey of Deterministic Project Scheduling," *Omega,* vol. 29, no. 3, pp. 249-272, 2001. [CrossRef] [Google Scholar] [Publisher Link]

[3] Rainer Kolisch, and Sönke Hartmann, "Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling: An Update," *European Journal of Operational Research*, vol. 174, no. 1, pp. 23-37, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[4] Sönke Hartmann, "A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling," *Naval Research Logistics (NRL),* vol. 45, no. 7, pp. 733-750, 1998. [CrossRef] [Google Scholar] [Publisher Link]

[5] Sönke Hartmann, "A Self-Adapting Genetic Algorithm for Project Scheduling Under Resource Constraints," *Naval Research Logistics (NRL),* vol. 49, no. 5, pp. 433-448, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[6] Kwan Woo Kim, Mitsuo Gen, and Genji Yamazaki, "Hybrid Genetic Algorithm with Fuzzy Logic for Resource-Constrained Project Scheduling," *Applied Soft Computing*, vol. 2, no. 3, pp. 174-188, 2003. [CrossRef] [Google Scholar] [Publisher Link]

[7] U. Kohlmorgen, H. Schmeck, and K. Haase, "Experiences with Fine-Grainedparallel Genetic Algorithms," *Annals of Operations Research*, vol. 90, pp. 203-219, 1999. [CrossRef] [Google Scholar] [Publisher Link]

[8] Jae-Kwan Lee, and Yeong-Dae Kim, "Search Heuristics for Resource Constrained Project Scheduling," *Journal of the Operational Research Society*, vol. 47, no. 5, pp. 678-689, 1996. [CrossRef] [Google Scholar] [Publisher Link]

[9] V. Jorge Leon, and Ramamoorthy Balakrishnan, “Strength and Adaptability of Problem-Space Based Neighborhoods for Resource-Constrained Scheduling,” *Operations-Research-Spektrum*, vol. 17, pp. 173-182, 1995. [CrossRef] [Google Scholar] [Publisher Link]

[10] Jorge Magalhães-Mendes, Jose F. Gonçalves, and MaurÃ-cio G. C. Resende, “A Random Key Based Genetic Algorithm for the Resource Constrained Project Scheduling Problem,” *Computers and Operations Research*, vol. 36, no. 1, pp. 92-109, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[11] Vicente Valls, Francisco Ballestín, and Sacramento 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. [CrossRef] [Google Scholar] [Publisher Link]

[12] Reza Zamani, “A Competitive Magnet-Based Genetic Algorithm for Solving the Resource-Constrained Project Scheduling Problem,” *European Journal of Operational Research*, vol. 229, no. 2, pp. 552-559, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[13] Kamel Bouleimen, and H. Lecocq, “A New Efficient Simulated Annealing Algorithm for the Resource-Constrained Project Scheduling Problem and its Multiple Mode Version,” *European Journal of Operational Research*, vol. 149, no. 2, pp. 268-281, 2003. [CrossRef] [Google Scholar] [Publisher Link]

[14] Fayez Fouad Boctor, *An Adaptation of the Simulated Annealing Algorithm for Solving Resource-Constrained Project Scheduling Problems*, Laval University, Research Directorate, Faculty of Administration, 1994. [Google Scholar] [Publisher Link]

[15] Daniel Merkle, Martin Middendorf, and Hartmut Schmeck, “Ant Colony Optimization for Resource-Constrained Project Scheduling,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 333-346, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[16] Shih-Tang Lo et al., “Multiprocessor System Scheduling with Precedence and Resource Constraints Using an Enhanced ant Colony System,” *Expert Systems with Applications*, vol. 34, no. 3, pp. 2071-2081, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[17] Hong Zhang, Heng Li, and C. M. Tam, “Particle Swarm Optimization for Resource-Constrained Project Scheduling,” *International Journal of Project Management*, vol. 24, no. 1, pp. 83-92, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[18] Qiong Jia and Yoonho Seo, “Solving Resource-Constrained Project Scheduling Problems: Conceptual Validation of FLP Formulation and Efficient Permutation-Based ABC Computation,” *Computers & Operations Research*, vol. 40, no. 8, pp. 2037-2050, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[19] Rainer Storn, and Kenneth Price, “Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces,” *Journal of Global Optimization*, vol. 11, pp. 341-359, 1997. [CrossRef] [Google Scholar] [Publisher Link]

[20] Najeh Damak et al., “Differential Evolution for Solving Multi-Mode Resource-Constrained Project Scheduling Problems,” *Computers and Operations Research*, vol. 36, no. 9, pp. 2653-2659, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[21] Jarboui Bassem et al., “A Combinatorial Particle Swarm Optimization for Solving Multi-Mode Resource-Constrained Project Scheduling Problems,” *Applied Mathematics and Computation*, vol. 195, no. 1, pp. 299-308, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[22] Amir Rahimi, Hamid Karimi, and Behrouz Afshar-Nadjafi, “Using Meta-Heuristics for Project Scheduling under Mode Identity Constraints,” *Applied Soft Computing*, vol. 13, no. 4, pp. 2124-2135, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[23] Jacek Blazewicz, Jan Karel Lenstra, and Alexander H. G. Rinnooy Kan, “Scheduling Subject to Resource Constraints: Classification and Complexity,” *Discrete Applied Mathematics*, vol. 5, no. 1, pp. 11-24, 1983. [CrossRef] [Google Scholar] [Publisher Link]