Journal of Theoretical and Applied Information Technology
31st May 2025. Vol.103. No.10
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

USING VARIANTS OF GENETIC ALGORITHM AND LEARNABLE EVOLUTION MODEL TO SOLVE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM

GAMAL ALSHORBAGY[1,2], MOHAMED EL-DOSUKY[1,3]

[1]Computer Science Department, Arab East Colleges, Saudi Arabia

[2]Climate Change Information Centre, Renewable Energy & Expert Systems, Giza, Egypt

[3]Computer Science Department, Faculty of Computers and Information, Mansoura University, Egypt

E-mail: maldosuky@arabeast.edu.sa

ABSTRACT

The resource-constrained project scheduling problem (RCPSP) is a complex scheduling challenge as it is proven to be NP-hard. The Learnable Evolution Model (LEM) is a non-Darwinian evolutionary approach that speeds up convergence using machine learning instead of crossover. It classifies individuals into high-performance (H-group) and low-performance (L-group) based on fitness, learns distinguishing features, and generates new individuals through an instantiation step. To ensure diversity, LEM applies mutation as a Darwinian component, making it more efficient than traditional evolutionary methods. This paper proposes a new approach, which attempts variants of genetic algorithms and LEM, aiming to tackle issues in generating Gantt charts for big cases.

Keywords: *Scheduling, RCPSP, Learnable evolution model, Genetic Algorithm*

1. INTRODUCTION

The resource-constrained project scheduling problem (RCPSP) is a complex scheduling challenge as it is proven to be NP-hard [1]. To tackle it effectively, simple, intuitive rules-of-thumb are often applied, providing sufficiently good and practical solutions, especially for large real-life cases [2].

The Learnable Evolution Model (LEM) is a non-Darwinian evolutionary computation approach that accelerates convergence using machine learning instead of traditional genetic operations like crossover [3-6]. Unlike genetic algorithms (GA), LEM classifies individuals into high-performance (H-group) and low-performance (L-group) based on fitness. A learning step then derives distinguishing characteristics between these groups. Using these learned descriptions, the instantiation step generates new individuals.

To maintain diversity, LEM incorporates mutation as a Darwinian component. This learning-driven process allows LEM to evolve solutions more efficiently than standard evolutionary methods.

This paper proposes a new approach, which attempts variants of GA and LEM, aiming to tackle issues in generating Gantt charts for big cases.

The rest of this paper is divided as follows. Section 2 provides a literature review on solving RCPSP using genetic algorithm. Then the proposed methodology is presented in a subsequent section, followed by a dedicated section for test cases, before concluding the paper.

2. LITERATURE REVIEW

Table 1 summarizes the literature review. Kaiafa and Chassiakos address resource allocation optimization in project management, focusing on multi-objective resource-constrained scheduling [7]. This work minimizes costs due to resource overallocation, deadline exceedance, and resource fluctuations. A genetic algorithm evaluates various activity execution alternatives, providing more balanced and effective solutions compared to Microsoft Project.

Another paper addresses a multi-project environment with due dates, alternative resource usage modes, and a non-shared resource dedication policy [8]. It discusses budget allocation, resource dedication to projects, and multi-mode resource-constrained scheduling (MRCPSP). Two solution approaches, a two-phase and a monolithic genetic algorithm, are proposed and evaluated through computational testing.

Mathew, Paul, Dileeplal, and Mathew develop a scheduling method for repetitive projects, aiming to minimize duration, cost, and both, while considering precedence relationships and due date constraints [9].

Yassine, Mostafa, and Browning introduce two GAs approaches for scheduling product development projects with resource and precedence constraints, aiming to minimize overall duration [10]. Tested with and without stochastic feedback, the GAs outperform 31 priority rules (PRs) in various conditions, providing managers with a decision matrix for choosing between GAs and PRs.

Eid, Elbeltagi, and El-Adaway present a multi-criteria optimization model using GAs and Pareto Front sorting for scheduling linear infrastructure projects, offering planners non-dominated alternatives and minimizing duration, cost, interruptions, and delays [11].

A probabilistic selection method has been developed to ensure diverse chromosome selections in resource provisioning and allocation [12].

Another paper introduces a multi-objective optimization method for repetitive scheduling, addressing time, cost, and work interruptions while accounting for uncertainties [13]. It integrates six modules, including fuzzy set theory and GA. The method optimizes project duration, cost, and interruptions, producing results close to those in the literature and accelerating schedule generation.

Another paper presents a schedule optimization framework for repetitive projects with scattered sites, such as bridge or school rehabilitations [14]. The framework includes a scheduling algorithm, cost optimization, interactive documentation, and legible schedule representation. A computer prototype was validated on real-life cases, demonstrating flexibility in scheduling without extra costs and supporting efficient infrastructure planning.

A recent study presents a new metaheuristic algorithm for solving resource-constrained multi-project scheduling [15]. The algorithm combines evolutionary operators and resource-buffered scheduling tactics. It outperforms existing methods on a large benchmark dataset, achieving the best-known solutions for 20% of test cases. A new metric is also introduced to analyze solution structures.

Although numerous studies have explored resource allocation optimization in project management, several limitations persist. Existing approaches, such as those by Kaiafa and Chassiakos, and others focusing on single or multi-project environments, often rely on genetic algorithms and deliver promising results. However, they tend to oversimplify real-world conditions by assuming static resources, deterministic scheduling, or limited project interactions. Many fail to incorporate uncertainty, dynamic changes, or real-time adaptability, which are critical in modern project portfolios. Some methods, while addressing multi-objective concerns like time and cost, lack scalability, generalizability, or practical decision-support features. Others focus narrowly on specific project types such as infrastructure or repetitive projects, limiting their broader applicability. Despite algorithmic advancements, few solutions provide interpretable, interactive tools for planners managing complex, dynamic, and interdependent projects. This highlights the need for a comprehensive, uncertainty-aware, and multi-objective optimization framework that can adapt to dynamic resource constraints, integrate practical decision-making, and remain applicable across diverse project scenarios.

3. NEWLY PROPOSED SYSTEM

As shown in Algorithm 1, the genetic algorithm begins by reading the project instance data as input and initializes a population of potential solutions. The algorithm then evaluates the fitness of this population. It proceeds by repeatedly performing crossover and mutation operations on the population until a specified stopping criterion is met. During each iteration, the population is ranked, and the best solutions are retained while lesser-performing ones are discarded. Finally, the algorithm concludes by selecting the incumbent solution as the best found during the optimization process. The overall goal is to evolve the population over generations to find an optimal or near-optimal solution to the problem at hand.

Algorithm 2 is a variant of Genetic Algorithm GA-based approach for optimizing project scheduling, where the goal is to minimize the project's makespan while considering constraints like task dependencies and durations. It begins by initializing a population of random schedules. Each schedule is evaluated using Forward-Backward Scheduling to compute the initial makespan. Then, Shortest Job First (SJF) Scheduling is applied to refine task sequencing. The fitness of each schedule is assessed based on makespan and constraint satisfaction. The algorithm then selects the best-performing schedules as parents and applies crossover to create offspring, followed by mutation to introduce genetic diversity. The offspring are evaluated, ranked, and the population is pruned to maintain diversity while improving the best-found solution. The process repeats until a stopping criterion is met, such as reaching a maximum number of generations or observing no further improvement in fitness.

Figure 1 is the proposed class diagram that represents a project scheduling system involving three main classes: Project, Individual, and Node. The Project class manages a collection of Node objects, stored in a dictionary, and a population of Individual objects. It includes methods for reading project data from a file and solving scheduling problems using a genetic algorithm.

The Individual class represents potential solutions with a fitness value and an activity_list of Node objects. The Node class models activities with attributes such as predecessors, successors, duration, and name, allowing for a directed graph structure. These classes interact closely, with the Project coordinating nodes and individuals, and the Individual using Node objects to form activity sequences for optimization.

In this paper, a new class of evolutionary computation processes is presented, called Learnable Evolution Model or LEM. As shown in figure 2, the flowchart that depicts the LEM algorithm contains a number of terms need to be defined. These terms are:

1. ***Fitness function:*** a method for evaluating performance of individuals in evolving populations. It assigns a quantitative or qualitative value to each individual. It represents a precondition for the LEM application.
2. ***learn-probe and dar-probe parameters:*** control persistence in continuing a given mode; specifically, they define the number of generations in Machine Learning and Darwinian Evolution modes, respectively, which are performed despite an unsatisfactory progress of the evolution process in the given mode.
3. ***learn-threshold and dar-threshold parameters:*** constitutes unsatisfactory evolutionary. These parameters specify threshold ratios of the best fitness function value in the sequence of learn-probe and dar-probe populations to the best fitness in the previous sequence of populations in Machine Learning and Darwinian Evolution mode, respectively.
4. ***ML-TC:*** The Machine Learning mode termination condition is met when a plateau of performance is reached.
5. ***LEM-TC:*** LEM termination condition.
6. ***UniLEM:*** a version of LEM in which LEM always chooses the start-over operation, the evolution process is based solely on a repetitious application of Machine Learning mode.
7. ***DuoLEM:*** LEM version that applies two separate modes ML and DE.
8. ***Start over*** can be done using some alternative methods such as:

A. *Select-elite.* Randomly generate individuals but include them in the new population only if their fitness value is above a certain threshold.

B. *Use-recommendations.* This method requires storing a set of past H-group descriptions whose instantiations led to a jump in the maximal-so-far fitness in the evolution process. Such descriptions are called *consequential* and a rapid increase of the top fitness value is called an *insight jump*. A new starting population is generated by instantiating one or more consequential descriptions.

C. *Generate-a-variant.* This method generates a new start-over population by making modifications to the individuals of the last population. Such modifications may be random mutations, or may be results of specialized description modification operators tailored for a given application domain.

9. ***ML-mode:*** any learning method that can generate descriptions discriminating between classes of individuals in a population can be employed (eg. Inductive logic programming system, INDUCE relational learning system), but AQ-learner (which use AQ-learning algorithm) is a most suitable learner to the LEM.
10. ***DE-mode:*** any conventional evolutionary algorithm can be potentially applied (e.g. a

genetic algorithm, an evolutionary strategy, or an evolutionary program).

The flowchart outlines an iterative optimization process that integrates machine learning and differential evolution techniques. It begins with generating a population of individuals, either randomly or based on predefined rules, and evaluating their fitness values. The ML-mode is then executed to classify the population into high-fitness (H-Group) and low-fitness (L-Group) individuals.

A hypothesis is generated by applying ML techniques to distinguish between these groups. A new population is then created either by applying specific rules or by generating individuals randomly. If the ML-threshold condition (ML-TC) is met—indicating that the best fitness values do not improve significantly—the algorithm proceeds to check for the learning termination condition (LEM-TC), which detects a performance plateau. If the LEM-TC is not met, the process loops back to refining the population. If both conditions are satisfied, the DE-mode is executed, involving mutation, crossover, and selection operators to generate a new population. The DE-threshold condition (DE-TC) is then evaluated, and if improvement stagnates, another LEM-TC check is performed. If LEM-TC is met, the process terminates; otherwise, the loop continues until an optimal solution is found.

4. RESULTS

Figure 3 represents a Gantt chart illustrating scheduled activities over time for a trivial scheduling as in Table 2. Each bar corresponds to an activity, labelled from Activity 0 to Activity 6, with its duration displayed along the timeline. Figure 4 shows the resulted Gantt chart for the Sim-120 case.

The key difference between the proposed approach and prior research lies in its usage of genetic algorithms and Learnable Evolution Model (LEM) and its focus on scalability and automation in generating Gantt charts for large-scale project scheduling problems. While previous studies primarily rely on standard or enhanced genetic algorithms—often static in design and manually tuned—this paper introduces a more adaptive and intelligent search mechanism. The LEM component allows the algorithm to learn from high-quality solutions during the search process, guiding future generations more effectively than purely stochastic methods. This results in faster convergence, better scalability, and more meaningful exploration of complex scheduling spaces. Additionally, unlike earlier works which either ignore visualization or use post-processing to generate schedules, this approach directly incorporates Gantt chart generation within the optimization loop, addressing both performance and usability. In summary, the novelty lies in combining evolutionary learning with practical output generation, targeting large, dynamic, and realistic project environments that were not adequately addressed in prior research.

5. CONCLUSION

This paper proposed a new approach, which attempts variants of genetic algorithms and LEM, aiming to tackle issues in generating Gantt charts for big cases.

The primary objective of this research was to develop an intelligent and scalable scheduling approach that uses variants of GAs with a LEM to address the complexity of generating optimized Gantt charts for large-scale, resource-constrained, multi-project environments. The approach aimed to overcome the limitations of traditional metaheuristics by introducing a learning component that enhances search efficiency and solution quality, particularly under uncertainty and dynamic project conditions. Additionally, the integration of automated Gantt chart generation aimed to bridge the gap between optimization and practical, user-oriented visualization.

The extent to which these objectives were achieved is significant. The propsed framework demonstrated superior performance over conventional GAs and existing benchmarks, particularly in large test cases where traditional methods struggled with convergence and diversity. The model also proved effective in maintaining solution quality while generating visually interpretable Gantt charts in real time, validating its applicability in real-world project management scenarios.

However, some limitations and threats to validity must be acknowledged. First, the learning mechanism within LEM relies on patterns derived from previous generations, which might lead to premature convergence if diversity is not adequately maintained. Second, the approach was primarily tested in controlled simulation environments, and further validation with real project datasets is necessary to confirm generalizability. Additionally, the computational complexity of using GAs and LEM may pose a challenge for extremely time-sensitive applications unless optimized through parallel computing or algorithmic simplifications. Despite these limitations, the proposed framework marks a significant step toward more intelligent and scalable project scheduling solutions.

6. FUTURE WORK

Future research should focus on enhancing the LEM by trying other machine learning algorithms [16]. Another future direction may try hybridizing the proposed model with other metaheuristic optimizers such as Particle Swarm Optimizer [17]. A future direction may attempt to solve stochastic version of RCPSP [18].

Recently, a set of measures for assessing the performance of RCPSP solutions [19]. This could be a good research direction to investigate the results in the light of these measures.

CODE AVAILABILITY:

The code that supports the findings of this paper is available from author Mohamed Eldosuky, upon request.

REFERENCES:

[1] Blazewicz, J., Lenstra, J. K., & Kan, A. R. (1983). Scheduling subject to resource constraints: Classification and complexity. Discrete Applied Mathematics, 5(1), 11–24

[2] Luo, J., Vanhoucke, M., Coelho, J., & Guo, W. (2022). An efficient genetic programming approach to design priority rules for resource-constrained project scheduling problem. Expert Systems with Applications, 198, 116753

[3] Michalski, R.S.: Learnable evolution: Combining symbolic and evolutionary learning. In: Proceedings of the Fourth International Workshop on Multistrategy Learning (MSL'98). (1998) 14–20

[4] Wojtusiak, J., Michalski, R.S.: The lem3 implementation of learnable evolution model and its testing on complex function optimization problems. In: GECCO. (2006) 1281–1288

[5] Michalski, R.S.: Learning and evolution: An introduction to non-darwinian evolutionary computation. In: ISMIS. (2000) 21–30

[6] Michalski, R.S.: Learnable evolution model: Evolutionary processes guided by machine learning. Machine Learning 38(1-2) (2000) 9–40

[7] Kaiafa, S., & Chassiakos, A. P. (2015). A genetic algorithm for optimal resource-driven project scheduling. Procedia Engineering, 123, 260-267.

[8] Beşikci, U., Bilge, Ü., & Ulusoy, G. (2015). Multi-mode resource constrained multi-project scheduling and resource portfolio problem. European Journal of Operational Research, 240(1), 22-31.

[9] Mathew, J., Paul, B., Dileeplal, J., & Mathew, T. (2016). Multi objective optimization for scheduling repetitive projects using GA. Procedia Technology, 25, 1072-1079.

[10] Yassine, A. A., Mostafa, O., & Browning, T. R. (2017). Scheduling multiple, resource-constrained, iterative, product development projects with genetic algorithms. Computers & industrial engineering, 107, 39-56

[11] Eid, M. S., Elbeltagi, E. E., & El-Adaway, I. H. (2021). Simultaneous multi-criteria optimization for scheduling linear infrastructure projects. International Journal of Construction Management, 21(1), 41-55

[12] Samuel, B., & Mathew, J. (2018, March). Resource allocation in a repetitive project scheduling using genetic algorithm. In IOP Conference Series: Materials Science and Engineering (Vol. 330, No. 1, p. 012098). IOP Publishing.

[13] Salama, T., & Moselhi, O. (2019). Multi-objective optimization for repetitive scheduling under uncertainty. Engineering, Construction and Architectural Management, 26(7), 1294-1320.

[14] Hegazy, T., & Kamarah, E. (2022). Schedule optimization for scattered repetitive projects. Automation in Construction, 133, 104042.

[15] Bredael, D., & Vanhoucke, M. (2024). A genetic algorithm with resource buffers for the resource-constrained multi-project scheduling problem. European Journal of Operational Research, 315(1), 19-34

[16] Guo, Weikang, Mario Vanhoucke, José Coelho, and Jingyu Luo. "Automatic detection of the best performing priority rule for the resource-constrained project scheduling problem." Expert systems with applications 167 (2021): 114116.

[17] Zhang, Hong, Heng Li, and C. M. Tam. "Particle swarm optimization for resource-constrained project scheduling." *International journal of project management* 24, no. 1 (2006): 83-92.

[18] Choi, Jaein, Matthew J. Realff, and Jay H. Lee. "AQ-Learning-based method applied to stochastic resource constrained project scheduling with new project arrivals." International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal 17, no. 13 (2007): 1214-1231.

[19] Van Eynde, Rob, Mario Vanhoucke, and José Coelho. "On the summary measures for the resource-constrained project scheduling problem." Annals of Operations Research 337, no. 2 (2024): 593-625.

Table 1: Summary of literature review

Reference	Minimization objective	Number of Activities
[7]	Resource overallocation and project deadline	10
[8]	Total weighted project tardiness	120, 180
[9]	Project duration and cost	20, 90
[10]	Project duration	30, 60, 90, 120
[11]	Project duration, cost and interruptions	20, 75
[12]	Project duration	20
[13]	Project duration, cost and interruptions	20
[14]	Total project cost	30, 70
[15]	Total portfolio makespan and project delay	360, 720, 1440

Algorithm 1: Traditional Genetic Algorithm

```
Input: Project instance data
Output: Best solution found
Start
    Read project instance data.
    Initialize the population.
    Evaluate the fitness of the population.
    Repeat until stopping criteria are met:
        Perform Crossover
        Perform Mutation
        Rank and reduce the population
        Select the incumbent solution
End
```

Algorithm 2: A variant of Genetic Algorithm

```
Input: Project data (tasks, dependencies, durations, ...)
Output: Optimized project schedule
Start
    Initialize the population with random schedules.
    Repeat until stopping criteria are met:
        Apply Forward-Backward Scheduling to calculate initial
```

```
          makespan
          For each individual in the population:
                    Apply Shortest Job First Scheduling
                    Evaluate the fitness of the schedule based on
                    makespan and project constraints
          Select parents based on fitness
          Apply Crossover to generate offspring
          Apply Mutation to introduce genetic diversity
          Evaluate the fitness of offspring schedules
          Rank and reduce the population to maintain diversity and
          improve the best solution
          If generation reaches maximum or incumbent fitness improves,
          stop
End
```

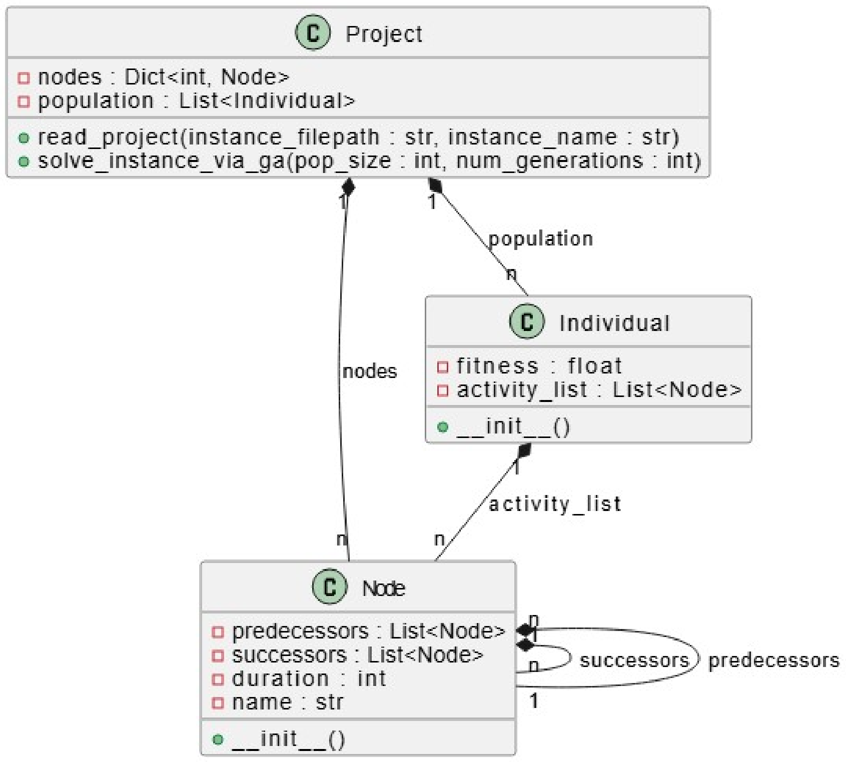
Project
nodes : Dict<int, Node>
population : List<Individual>
read_project(instance_filepath : str, instance_name : str)
solve_instance_via_ga(pop_size : int, num_generations : int)
1
1
population
n
nodes
Individual
fitness : float
activity_list : List<Node>
__init__()
1
activity_list
n
n
Node
predecessors : List<Node>
successors : List<Node>
duration : int
name : str
__init__()
n
n
successors
predecessors
1

Figure 1: Proposed Class Diagram

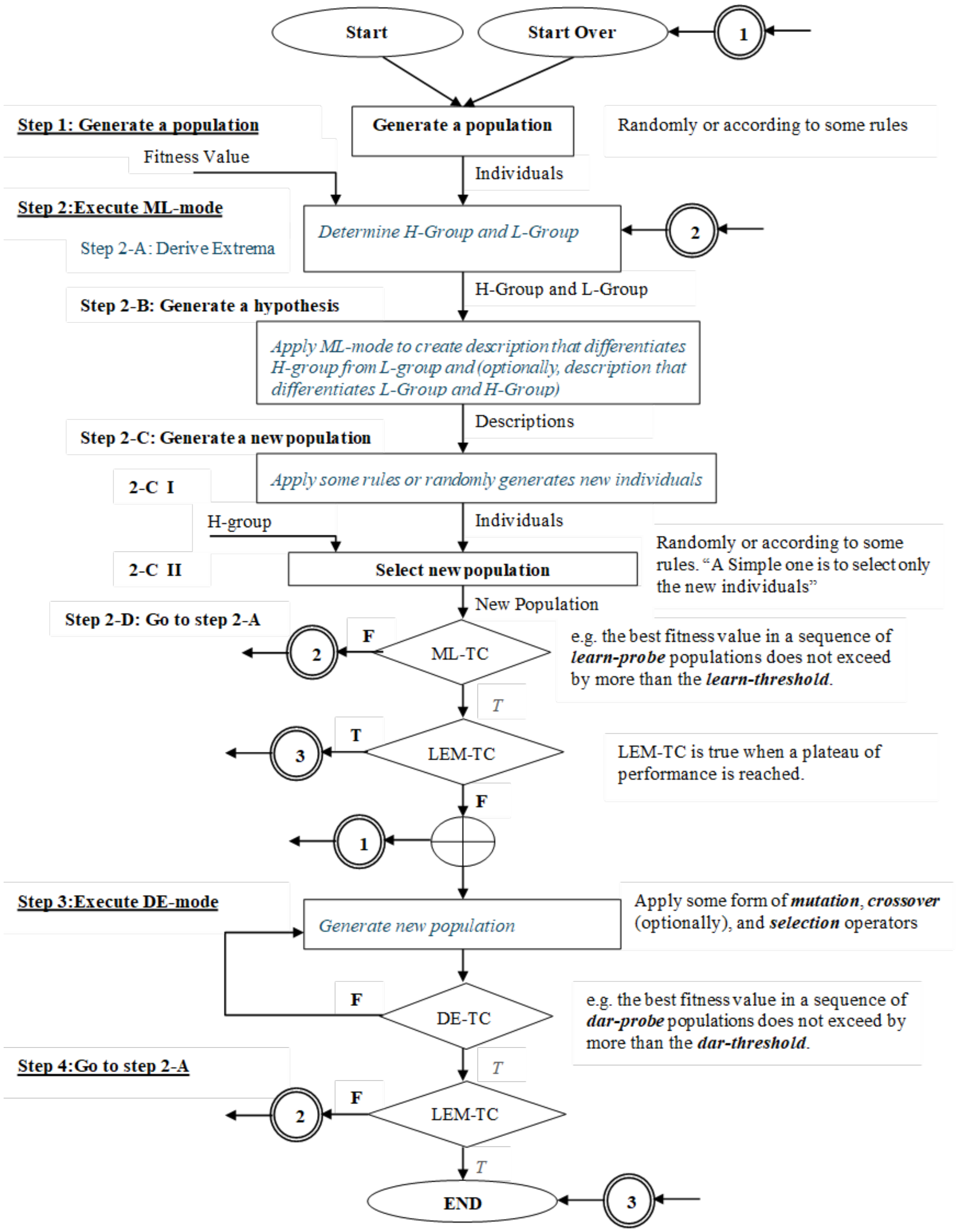
Start
Start Over
1
Step 1: Generate a population
Generate a population
Randomly or according to some rules
Fitness Value
Individuals
Step 2:Execute ML-mode
Determine H-Group and L-Group
2
Step 2-A: Derive Extrema
H-Group and L-Group
Step 2-B: Generate a hypothesis
Apply ML-mode to create description that differentiates H-group from L-group and (optionally, description that differentiates L-Group and H-Group)
Descriptions
Step 2-C: Generate a new population
Apply some rules or randomly generates new individuals
2-C I
H-group
Individuals
Randomly or according to some rules. "A Simple one is to select only the new individuals"
2-C II
Select new population
New Population
Step 2-D: Go to step 2-A
F
2
ML-TC
e.g. the best fitness value in a sequence of learn-probe populations does not exceed by more than the learn-threshold.
T
T
3
LEM-TC
LEM-TC is true when a plateau of performance is reached.
F
1
Step 3:Execute DE-mode
Generate new population
Apply some form of mutation, crossover (optionally), and selection operators
F
DE-TC
e.g. the best fitness value in a sequence of dar-probe populations does not exceed by more than the dar-threshold.
Step 4:Go to step 2-A
T
F
2
LEM-TC
T
END
3

Figure 2: LEM Algorithm

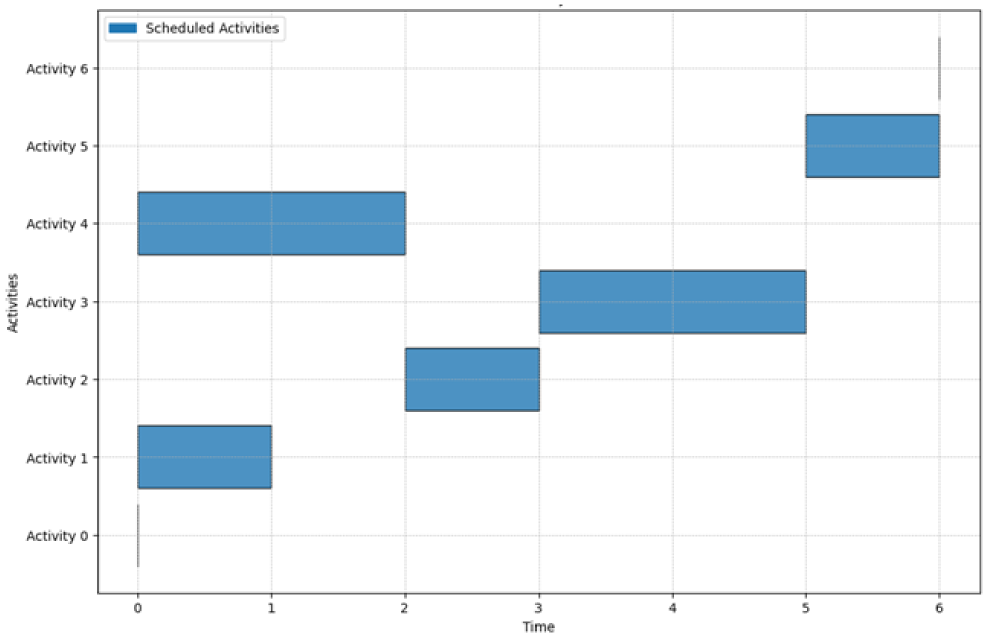

Figure 3: Gantt chart for a Trivial Case

Table 2: Trivial Example for scheduling

Activity	Start Time	End Time	Duration
Activity 0	0	1	1
Activity 1	2	3	1
Activity 2	3	4	1
Activity 3	3	5	2
Activity 4	1	3	2
Activity 5	5	6	1
Activity 6	5	6	1

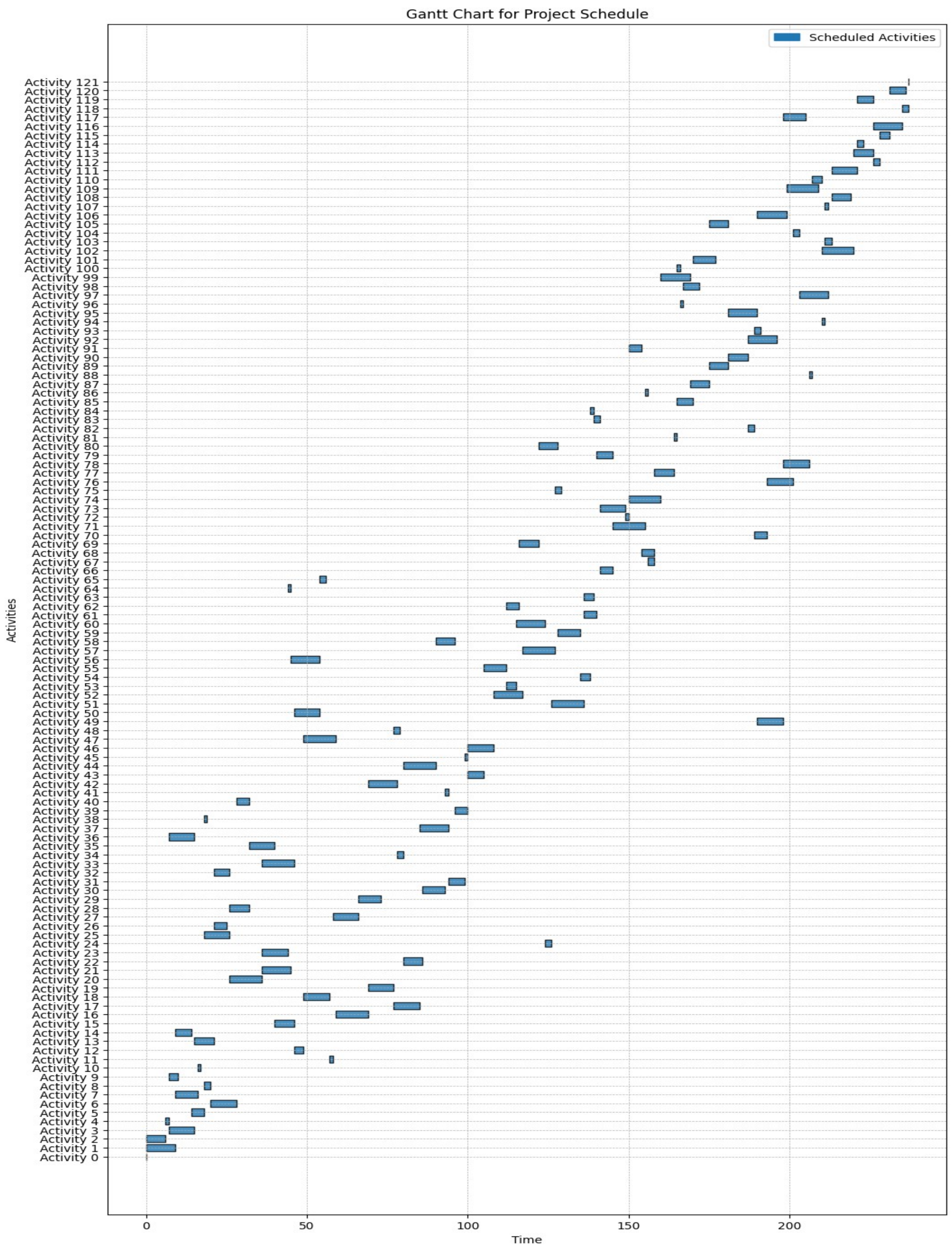

Figure 4: Gantt chart for the Sim-120 case