Journal of Engineering and Technology for Industrial Applications

ISSN ONLINE: 2447-0228

ITEGAM-JETIA

Manaus, v.12 n.58, p. 209-223. March/April, 2026.

DOI: https://doi.org/10.5935/jetia.v12i58.2982

RESEARCH ARTICLE OPEN ACCESS

MODELLING, VERIFICATION, AND OPTIMIZATION OF PROCESS PLAN GENERATION IN SUSTAINABLE RECONFIGURABLE MANUFACTURING SYSTEMS USING EVOLUTIONARY-GENERATED PETRI NETS

Mohamed Elkabir Fareh[*1], Laid Kahloul[2], Manel Femmam[3]

[1,2] Computer Science Department, Faculty of Sciences, Biskra University, Biskra, Algeria
[3]Computer Science Department, University Center of Barika, Barika, Algeria

[1]https://orcid.org/0009-0008-7308-4635, [2]https://orcid.org/0000-0002-9739-7715, [3]https://orcid.org/0009-0009-8536-1326

E-mail: *fareh.me@gmail.com; l.kahloul@univ-biskra.dz; manel.femmam@cu-barika.dz

ARTICLE INFO

Article History
Received: December 9, 2025
Reviewed: January 1, 2026
Accepted: January 13, 2026
Published: March 31, 2026

Keywords:
Sustainable Reconfigurable manufacturing systems,
Petri Nets,
Genetic algorithms,
Multi-objective optimization,
Formal verification.

ABSTRACT

Various evolutionary algorithms have been developed to address environmentally oriented multi-objective process planning challenges in reconfigurable manufacturing systems, though many prioritize effectiveness over flexibility. Research utilizing Petri nets effectively addresses this flexibility issue, with several extensions proposed for modeling complex systems. By leveraging established Petri net theory, the process plan of reconfigurable manufacturing systems (RMS) can be encoded, enabling the optimization of process plan generation and the identification of potential deadlocks. This paper introduces a novel approach to enhance process plan generation, combining Evolutionary Petri Nets (a variant of Petri nets) with the enhanced genetic algorithm N˜SGA-III. The method optimizes four key objectives within the context of Sustainable Reconfigurable Manufacturing Systems (SRMS): total production cost, total production time, greenhouse gas emissions from energy consumption, and hazardous liquid waste generation. Numerical experiments were conducted to validate the effectiveness of this approach.

I. INTRODUCTION

I.1 CONTEXT AND MOTIVATIONS

Reconfigurable Manufacturing System is a dynamic research domain that encompasses various advanced topics, including design, layout optimization, reconfigurable control, process planning, and production scheduling. However, there remains a significant gap in integrating sustainability into RMS studies [1]. A Sustainable Reconfigurable Manufacturing System (SRMS) is defined as a manufacturing system designed to be flexible and adaptable to shifts in product demand and production requirements, Additionally, SRMS focuses on minimizing environmental impacts while enhancing social and economic benefits [2]. Process planning, also known as process plan generation, is a critical initial step in organizing a manufacturing facility. It outlines how raw materials will be transformed into finished products. This planning occurs at two levels: macro and micro. Macro-process planning focuses on selecting the optimal sequence of various processing steps, setups, and the machines needed for the operations. In contrast, micro-process planning aims to optimize each individual operation to identify the best process parameters. Therefore, process planning is essential within the entire manufacturing cycle. Both levels of planning present complex challenges, attracting attention from both industry professionals and researchers, leading to numerous contributions in the field. Process plan generation using Petri nets is an effective method for modeling and analyzing manufacturing processes. Petri nets provide a graphical and mathematical representation that captures the dynamics of systems, allowing for the visualization of workflow, resource allocation, and concurrent activities. Evolutionary algorithms or computations (EC) are commonly used for solving problems such as optimization and reverse engineering of complex systems [2]. Genetic Algorithms (GAs) are widely used to solve various manufacturing problems, particularly in the context of RMS, due to their ability to effectively handle complex, multi-dimensional search spaces.

In these environments, GAs help optimize diverse objectives, such as cost, time, and resource allocation, under dynamic and uncertain conditions. Among the multi-criteria versions of GAs, the most commonly employed is NSGA-II [3], known for its robust performance in finding Pareto-optimal solutions and maintaining diversity in the solution set, making it suitable for complex decision-making scenarios in manufacturing systems. The integration of evolutionary algorithms and Petri nets provides a powerful method for generating and optimizing process plans in manufacturing systems. This approach enhances the ability to address complex challenges, especially in environments that prioritize sustainability and efficiency. These benefits motivated us to propose in this paper an Evolutionary-Generated Petri Nets based approach for Process Plan Generation in Sustainable Reconfigurable Manufacturing Systems.

I.2 KEY CONTRIBUTIONS

Our research paper seeks to develop a multi-objective sustainable process plan generation at a macro level within a reconfigurable manufacturing environment through:

- A formal modeling of the multi-objective sustainable process plan generation problem in RMS, considering production costs, completion times, greenhouse gas emissions, and hazardous liquid waste as optimization criteria.
- An approach based on Evolutionary-Generated Petri Nets (EPN) to effectively solve the sustainable process plan generation challenge in RMS.
- The identification of potential deadlocks in the process plan by utilizing the reachability graph (RG) of a Petri Net.
- An investigation into how the probabilities of genetic operators affect the convergence of the adapted NSGA-III algorithm.

The remainder of this paper is organized as follows: Section 2 provides a literature review, followed by the problem description in Section 3. The modeling of the process plan using evolutionary Petri nets is discussed in Section 4. Section 5 presents the proposed evolutionary approach, while Section 6 focuses on the verification of the process plan. Section 7 introduces the adapted evolutionary algorithm. The experimentation of the approach is detailed in Section 8, and Section 9 concludes with a discussion of future work.

II. LITERATURE REVIEW

Sustainable Reconfigurable Manufacturing Systems (SRMS) have gained significant attention across both academia and industry due to their critical role in flexible, eco-friendly production [4]. This section surveys key contributions that integrate artificial intelligence (AI) techniques—particularly multi-objective evolutionary optimization—into sustainable process planning within RMS and SRMS contexts. Several studies [5], [6], [7] have extensively investigate multi-objective process and production planning in SRMS by formulating models that explicitly incorporate the three pillars of sustainability: social, environmental, and economic. Their frameworks combine classical optimization techniques, including Lagrangian relaxation for large-scale problems and exact solution methods via GAMS for smaller instances, alongside linear mixed-integer programming and augmented ε-constraint approaches, demonstrating the importance of rigorous mathematical modeling in sustainability-focused manufacturing planning. In RMS, energy cost optimization is tackled through bi-level optimization frameworks that couple line balancing with production scheduling.

A hybrid metaheuristics that synergizes simulated annealing with linear programming has been proposed in [8] to solve such problems effectively. Significant contributions have been made in [9], [10], [11], [12] through the development of multi-objective, environmentally focused evolutionary algorithms aimed at optimizing sustainable machine and tool selection. These studies investigate advanced variants of NSGA-II and NSGA-III, incorporating techniques such as penalty boundary intersection and similarity coefficient-based operators to improve performance across key objectives, including production cost, processing time, greenhouse gas emissions, and hazardous waste generation. In [13], [14], several hybrid metaheuristics have been developed and evaluated—ranging from iterative multi-objective integer linear programming (I-MOILP) to archive-based iterated local search algorithms—comparing them against established evolutionary methods like AMOSA and NSGA-II. These studies highlight the effectiveness of hybrid AI approaches in solving complex, multi-objective manufacturing problems. Similarly, in [15], [16], multi-objective process planning has been addressed using mixed-integer linear programming models combined with evolutionary algorithms such as NSGA-II and SPEA-II.

Their contributions emphasize reducing completion time, production cost, and energy consumption within reconfigurable environments through intelligent algorithmic adaptations. This trend is extended by concurrently optimizing process planning, job-shop scheduling, and layout design within RMS in [17]. Their use of NSGA-III alongside well-defined economic and environmental metrics exemplifies state-of-the-art AI-enabled multi-objective optimization targeting sustainability improvements in manufacturing. Collectively, these studies demonstrate the growing role of evolutionary algorithms and hybrid metaheuristics in addressing the multi-faceted, sustainability-focused optimization challenges inherent to SRMS. However, there remains a critical need for integrated frameworks that combine formal modeling and verification, as well as adaptive evolutionary optimization—motivating the approach developed in this paper. **Table 1** presents an overview of recent studies addressing key issues in S-RMS problems.

Table 1: Summary of recent publication related to some S-RMS Problem.

Work	Optimisation methods	Process planning	Production planning	Job-shop scheduling	Time	Cost	Sustainability		
							GHG	Wastes	Energy
[5]	Lagrangian relaxation	✓	✓		✓	✓	✓	✓	
[6]	Adapted version of the Lp-metric	✓	✓			✓	✓	✓	
[7]	Augmented ε-constraint	✓	✓		✓	✓	✓	✓	
[8]	• Simulated annealing • Linear program		✓						✓

[9]	• Penalty Boundary Intersection Reference point with NSGA-II (PBI-R-NSGA-II) • NSGA-III with a selection and elimination operator (SE-NSGA-III) • NSGA-III with the similarity coefficient (SC-NSGA-III)	✓			✓	✓	✓	✓	
[10]	• NSGA-II • NSGA-III • Weighted genetic algorithms (WGA) • Random weighted genetic algorithms (RWGA)	✓			✓	✓	✓	✓	
[11]	• Dynamic-NSGA-II • New Dynamic-NSGA-II	✓			✓	✓	✓	✓	
[12]	• Adopted version of NSGA-II • Adopted version of NSGA III	✓			✓	✓	✓	✓	
[13]	• Iterative multi-objective integer linear programming (I-MOILP) • Archived Multi-Objective Simulated Annealing (AMOSA) • Adopted version of NSGA-II	✓			✓	✓	✓	✓	
[14]	• RSUPP: Repetitive single-unit process plan meta-heuristic • LSSUPP: Iterated local search on single-unit process plans meta-heuristic • ABILS: Archive-based iterated local search metaheuristic	✓			✓	✓	✓	✓	
[15]	• Augmented- ε-constraint (AUGECON) • NSGA-II • strength Pareto evolutionary algorithm II (SPEA-II)	✓			✓	✓	✓	✓	
[16]	• Augmented ε-constraint (AUGECON)	✓			✓	✓			✓
[17]	NSGA-III	✓		✓		✓	✓	✓	

Note: NSGA: Non-dominated Sorting Genetic Algorithm. GHG: Greenhouse gas emissions.
Source: Authors, (2026).

III. PROBLEM DESCRIPTION

Let us consider a single unit of a product to be manufactured in a reconfigurable environment. The product requires a set of operations, which are linked by precedence constraints, generally modelled by Directed Acyclic Graph (DAG): G = {V, E}, the set of vertices V = {OP_1,…, OP_n} denote each individual operation in the production process, E the set of directed edges represent precedence constraints between operations nodes. A directed edge $E_{i,j}$ states that OP_i is the parent operation of OP_j. Child operations can only be executed once all the parent operations are completed, as illustrated in Figure 1.

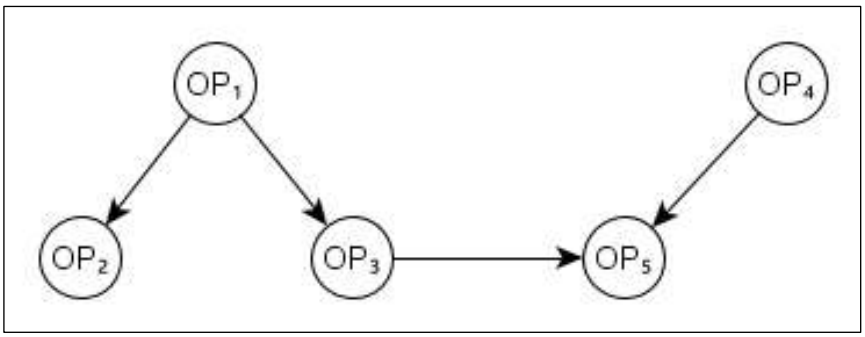

Figure 1: A simple operations precedence DAG.
Source: Authors, (2026).

An operation in this case is defined by a set of machines capable of performing it. A machine, on the other hand, is defined by a set of available configurations and a set of compatible tools. Table 2 presents configurations and tools that each machine offers. Thus,

each operation OP_i requires an association of machine-configuration-tool (M,C,T) called triplet TO_i. A feasible association between the operations and triplets is called process plan, which should respect the precedence constraints (Khettabi et al. 2021c).

Table 2: Example of configurations and tools that each machine offers.

M	C	T
M_1	C_1	T_1, T_2, T_3
M_2	C_2, C_3	T_1, T_3
M_3	C_1, C_2	T_2, T_3

Source: Authors, (2026).

Table 3: Example of a generated process plan.

Operation	OP_1	OP_4	OP_2	OP_3	OP_5
Machine	M_2	M_1	M_3	M_2	M_1
Configuration	C_3	C_1	C_2	C_2	C_1
Tool	T_1	T_2	T_2	T_3	T_1

Source: Authors, (2026).

More specifically, the process plan takes the structure of a matrix with n columns (number of operations) and four rows (operation, machine, configuration and tool). Table 3 shows an illustrative example of a process plan. It is read from left to right and column-by column. As an example, the first column indicates that: operation OP_1 is realised in the first step of the process plan, using machine M_2 under configuration C_3 with tool T_1.

IV. MODELLING PROCESS PLAN USING EVOLUTIONARY PETRI NETS

IV.1 MOTIVATION

The Directed Acyclic Graph (DAG) has a very simple structure and is easy to use. It possesses, however, some relevant disadvantages. As DAG is directed, it is impossible to define bidirectional coupling schemes between software components. DAG is acyclic, so it is not feasible to define loops, and only it describes the behaviour, but not the state of the system. These disadvantages motivate to use of the Petri nets formalism [18]. Petri nets are useful tool for visualisation of the processes. However, constructing and optimizing PN that describes process with hundreds of possible transitions and places is time and resources demanding task. Therefore, Evolutionary Petri Nets (EPN) are proposed as a solution to this task [2].

IV.2 FORMAL DEFINITION OF EPNs

Evolutionary Petri Nets (EPN) is proposed in [19] as a new extension for RPN (Resizable Petri Net). The aim of this proposition is to develop an evolutionary methodology whose candidate solutions are based on PNs, with two genetic operators, crossover and mutation, that altogether give a foundation for the development of a robust evolutionary algorithm for the optimisation. EPNs provide a conceptual framework for the representation of candidate solutions [18]. Below, we give the formal definition of RPNs and EPNs.

Definition 1: A Resizable Petri Net (RPN) [19] (extension of PNs) is defined as a 9-tuple: $\xi = (P, P^h, T, T^h, F, W, M_0, O_{pre}, O_{post})$:

- $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places;
- P^h is the set of hidden places, such that $P \cap P^h = \varnothing$;
- $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions, where: $P \cap T = \varnothing$ and $P \cup T \neq \varnothing$;
- T^h is the set of hidden transitions, such that $T \cap T^h = \varnothing$ and $(P \cup P^h) \cap (T \cup T^h) = \varnothing$;
- $F \subseteq ((P \cup P^h) \times (T \cup T^h)) \cup ((T \cup T^h) \times (P \cup P^h))$ is set of arcs;
- W: $F \rightarrow N$ is a weight function that associates a non-negative integer value to each arc;
- M_0: $P \rightarrow N$ is the initial marking of non-hidden places of the net (all hidden places have zero tokens, initially);
- $O_{pre} \in N$ is the maximum pre-order allowed in the RPN (equation 1), that is, for each $t \in (T \cup T^h)$: $\sum_{\substack{p \in Pre(t) \\ p \in (p \cup p^h)}} W(p, t) \leq O_{pre}$ (1)
- $O_{post} \in N$ is the maximum post-order allowed in the RPN (equation 2), that is, for each $t \in (T \cup T^h)$: $\sum_{\substack{p \in Post(t) \\ p \in (p \cup p^h)}} W(t, p) \leq O_{post}$ (2)

Definition 2: An Evolutionary Petri Net (EPN) [19] is a triple $E = (\xi, X, \mu)$, where:

- $\xi \in \Xi$ is a RPN; where Ξ is the space of all possible RPN topologies;
- X: $(\Xi \times \Xi) \rightarrow (\Xi \times \Xi)$ is the crossover operator which takes as input two RPNs ξ and $\bar{\xi}$ (where $P_\xi = P_{\bar{\xi}}$) and gives two new RPNs;
- μ: $\Xi \cup \{P_{in}, t, P_{out}\} \rightarrow \Xi$ is the mutation operation, where $\{P_{in}, t, P_{out}\}$ is a triple consisting of two places P_{in} and P_{out} and a transition t, namely $P_{in}, P_{out} \in (P \cup P^h) \cup P^\infty$ and $t \in (T \cup T^h) \cup T^\infty$, where P^∞ and T^∞ are infinite sets of places and transitions, such that $P^\infty \cap (P \cup P^h) = \varnothing$ and $T^\infty \cap (T \cup T^h) = \varnothing$.

Both the crossover and mutation operators are described in detail in [19].

IV.3 PROCESS PLAN MODELLING WITH RPNs

The process plan generation consists of sequencing the operations to be performed on the machines (under machines configurations, used tools, and precedence graph constraints) and the triplets to perform each operation in the sequence [10]. For modelling the process plan with RPNs, we introduce two kinds of places, the first represents machines, configurations, tools and the other represents dependencies from an operation OP_i to operation $OP_{i`}$. On the other hand, we introduce a type of transition represents set of available configurations and compatible tools for each machine.

In this work, Let us consider:
n: number of operations,
OP: set of operations $\{OP_1, OP_2, \dots, OP_n\}$,
m: number of machines,
M: set of machines $\{M_1, M_2, \dots, M_m\}$,
cfg: number of configurations,
C: set of configurations $\{C_1, C_2, \dots, C_{cfg}\}$,
tl: number of tools,
T: set of tools $\{T_1, T_2, \dots, T_{tl}\}$.

We can model a feasible process plan (candidate solution) with an RPN as a 8-tuple = $(P, P^h, T, T^h, F, W, M_0, O_{pre}, O_{post})$ where:

- $P^h = \{D_{ii`}\} \cup \{M_j\}$ where:

$\{D_{ii`}\}$ / $i,i` \in [1, n]$ representing the set of execution conditions of each operation (precedence constraints), $D_{ii`}$ denotes precedence constraint between operation $OP_i$ and $OP_{i`}$;

$\{M_j\}$ / $j \in [1, m]$ representing the set of machines;

- $P = \{T_t\} \cup \{C_c\}$ where :

$\{T_t\}$ / $t \in [1, tl]$ representing set of configurations;

$\{C_c\}$ / $c \in [1, cfg]$ representing set of tools;

Such that $P \cap P^h = \varnothing$

- $T = OP = \{OP_1, OP_2, \dots, OP_n\}$ representing set of operation composing production process, where: $P \cap T = \varnothing$ and $P \cup T \neq \varnothing$;
- $F \subseteq ((P \cup P^h) \times (T \cup T^h)) \cup ((T \cup T^h) \times (P \cup P^h))$ is set of arcs;
- W: $F \rightarrow$ N is a weight function that associates a non-negative integer value to each arc; in this case $W(f) = 1$, $\forall f \in F$;
- M_0: $P \rightarrow$ N is the initial marking of non- hidden places of the net (all hidden places have zero tokens, initially); this condition is respected in the present work excepted the starts places which are initially marked by one token;
- $O_{pre} \in$ N is the maximum pre-order allowed in the RPN (equation 1), that is, for each $t \in (T \cup T^h)$: $\sum_{\substack{p \in Pre(t) \\ p \in (p \cup p^h)}} W(p,t) \leq O_{pre}$ (3)
- $O_{post} \in$ N is the maximum post-order allowed in the RPN (equation 4), that is, for each $t \in (T \cup T^h)$: $\sum_{\substack{p \in Post(t) \\ p \in (p \cup p^h)}} W(t,p) \leq O_{post}$ (4)

Figure 2 depicts an RPN modelling a valid generated process plan (Table 3) for the operations precedence DAG (Figure 1).

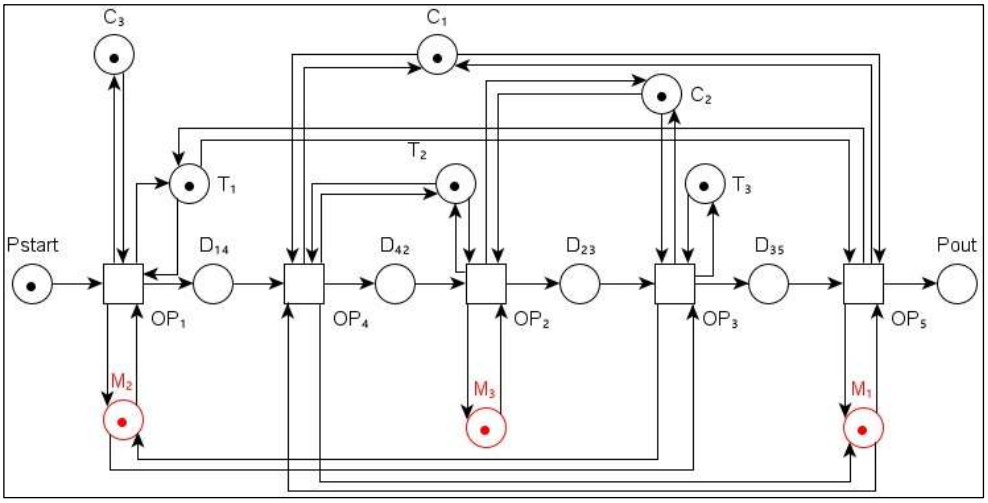

Figure 2: Modelling of valid generated process plan with RPN.
Source: Authors, (2026).

V. PROPOSED EVOLUTIONARY APPROACH

To identify the optimal process plan, we employ an evolutionary approach utilizing a genetic algorithm. Each potential chromosome or individual in our approach signifies a viable solution.

V.1 CHROMOSOME ENCODING

In the context of the process planning problem, a chromosome consists of a sequence of genes, with each gene representing the assignment of a specific operation in the process plan to its corresponding machine. During each transition firing, we assess key metrics of the process plan up to that transition, including makespan, production costs, hazardous liquid waste generation, and greenhouse gas emissions. Chromosome encoding is fundamentally based on the utilization of Evolutionary Petri Nets (see Definition 2). To achieve this, we employed the RPN model proposed in the previous section and assigned a label L = (***FT***, ***C***, **LHW**, ***GHG***) to each operation; where **FT** signify the finishing time of an operation, **C** signify the cost of execution of each operation OP_i on the selected machine M_j, **LHW** and **GHG** signify the amount of hazardous liquid waste and the amount of greenhouse gases emitted during the execution of each operation OP_i respectively. Formally, we define:

- EPN is the evolutionary Petri net introduced above.
- *FT: T x P → ℝ; (OPi, Mj) → FT (OPi, Mj)*

FT (OP_i, M_j) = FT (OP_i) = ST (OP_i) + $Pt_i + TCC_{c,\dot{c}} + TCM_{j,} + TCT_{t,\dot{t}}$

Where: ST (OP_i) = $\max_{p \in PR_i}$ {FT (OP_p)}

- *C: T x P → ℝ; (OPi, Mj) → C (OPi, Mj)*

C (OP_i, M_j) = C (OP_i) = Pc_i x $Pt_i + CCM_{j,}$ x $TCM_{j,} + CCC_{c,\dot{c}}$ x $TCC_{c,\dot{c}} + CCT_{t,\dot{t}}$ x $TCT_{t,\dot{t}}$

- *LWH: T x P → ℝ; (OPi, Mj) → LWH (OPi, Mj)*

LWH (OP_i, M_j) = LWH (OP_i) = L_i x Pt_i x EP_i

- *GHG: T x P → ℝ; (OPi, Mj) → GHG (OPi, Mj)*

GHG (OP_i, M_j) = GHG (OP_i) = IEC_j+ Pt_i x $f_{i,g}$ + Pe_i x $Pt_i + ECC_{c,\dot{c}}$ x $TCC_{c,\dot{c}}$ +$ECT_{t,\dot{t}}$ x $TCT_{t,\dot{t}} + ECM_{j,}$ x $TCM_{j,}$

The parameters used in these functions are as follows:

PR_i Set of predecessors of operation OP_i
G Set of greenhouse gases
$i, \dot{\imath}$: Index of operations OP_i and $OP_{\dot{\imath}}$
$j, \dot{\jmath}$: Index of machines M_j and $M_{\dot{\jmath}}$
$c, \dot{c}$ Index of configurations
$t, \dot{t}$ Index of tools
g Index of greenhouse gases
$TCM_{j,\dot{\jmath}}$ Machine change over time
$TCC_{c,\dot{c}}$ Configuration changeover time
$TCT_{t,\dot{t}}$ Tool changeover time
$CCM_{j,\dot{\jmath}}$ Machine change over cost per time unit
$CCC_{c,\dot{c}}$ Configuration changeover cost per time unit
$CCT_{c,\dot{c}}$ Tool changeover cost per time unit
$ECC_{c,\dot{c}}$ Configuration changeover energy per time unit
$ECT_{t,\dot{t}}$ Tool changeover energy
$ECM_{j,\dot{\jmath}}$ Machine changeover energy per time unit
Pt_i Operation OP_i processing time
Pc_i Operation OP_i processing cost
Pe_i Operation OP_i processing energy
$f_{i,g}$ Operation OP_i emitting greenhouse gas type g per time unit
IEC_j Initial energy consumption of machine M_j
EP_i Estimated hazardous liquid waste for operation OP_i
l_i Required liquid for operation OP_i per time unit

V.2 GENETIC OPERATORS

Genetic algorithm introduces crossover and mutation operations to increase the diversity of the population.

V.2.1 Crossover of RPN

The crossover mechanism between two RPNs generates new offspring which inherit the best substructures of the parents. No specific work exists about the crossover between bipartite graphs or, more specifically PNs. Indeed, the crossover between two PNs is supposed to identify some substructures in each graph, 'detach' them from one parent graph and 'attach' them into the other, and vice versa, while keeping the consistency of both graphs [19]. Our proposal for a crossover mechanism ($Crossover_{RPN}$) of two RPNs: $\xi(\tau), \bar{\xi}(\tau) \in \Xi$ considers four points:

- One operation (representing an operation in process plan problem) $OP_\chi \in OP$ is randomly selected in the $\xi(\tau)$ and the same operation is selected from $\bar{\xi}(\tau)$.

- Select the substructures consisting of the preset and post set of OP_χ (resp. $\overline{OP}_\chi$);
- Change the substructures between ξ and $\overline{\xi}$.

The use of this method saves the precedence constraints in each RPN.

V.2.2 Mutation of RPN

A mutation generates new RPN (offspring) from a single RPN (parent) in the current population. According to [13], the mutation operator modifies the structure of a RPN $\xi(\tau)$ in Ξ. We have proposed a mutation operator ($Mutation_{RPN}$) that associates $\xi(\tau)$ to a new consistent EPN $\xi(\tau + 1)$, according to specified couple $\{OP_i, M_j\}$. Our mutation operator acting on a single randomly chosen operation OP_i, the assignment of this operation will change as follows: the algorithm checks if there exists a machine with the same available configuration C_c and compatible tool T_t (denoted $M_{j'}$) unusable in all initial population ($M_{j'} \in P^{h\infty}$), this $M_{j'}$ will execute OP_i. If not ($P^{h\infty}=\varnothing$), the $M_{j'}$ must be randomly chosen from the list of $M_{j'}$ already used ($M_{j'} \in P^h$). Figure 3 a-b shows a sample of mutation operator. Assuming in this sample that $OP_\chi = OP_1$.

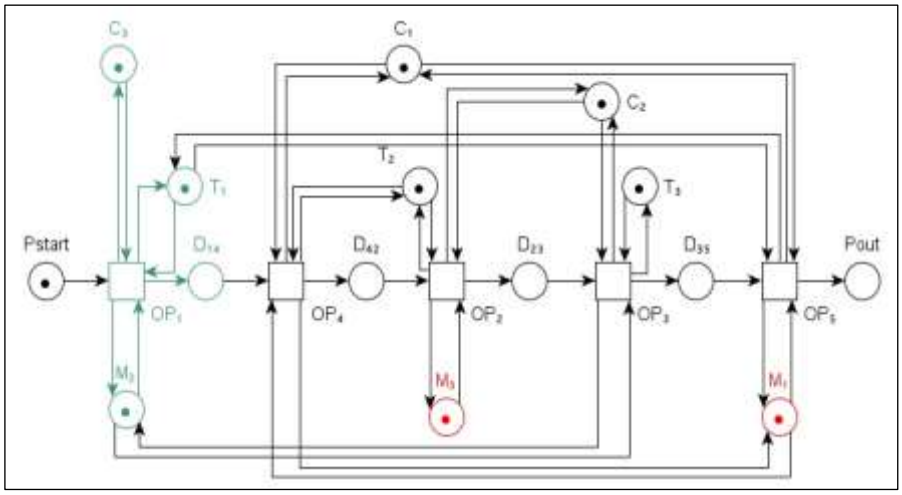

Figure 3 a: Befor mutation $\xi(\tau)$.
Source: Authors, (2026).

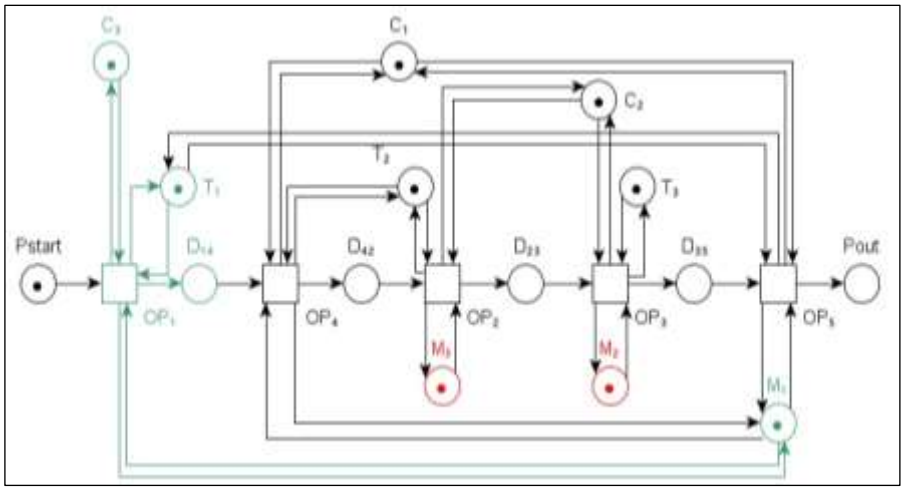

Figure 3 b: After mutation $\xi(\tau+1)$.
Source: Authors, (2026).

V.2.3 fitness function

The generated process plans must align with both manufacturing and sustainability objectives. Specifically, we aim to minimize four criteria:

- ✓ **Criterion 1: Total Production Cost** - This encompasses expenses related to machine changes, configuration adjustments, tooling, process alterations, as well as costs associated with greenhouse gas emissions and hazardous liquid waste.
- ✓ **Criterion 2: Total Production Time** - This includes the duration of machine transitions, configuration changes, tool replacements, and the time allocated for processing operations.
- ✓ **Criterion 3: Waste Generation** - This refers to hazardous liquid waste produced during operational processes, including waste oils and water, by products from resin production and application, residues from metal and plastic surface treatment, and materials resulting from industrial waste disposal activities.

✓ **Criterion 4: Greenhouse Gas Emissions** - This criterion captures the total emissions of greenhouse gases throughout the entire production process.

A comprehensive overview of these criteria, along with their mathematical formulations, is provided in [11]. By emphasizing these criteria, our goal is to create process plans that are both efficient and sustainable.

VI. VERIFICATION OF RTPM (VERIFICATION$_{RTPM}$)

Deadlock and livelock problems can lead to disastrous outcomes in reconfigurable manufacturing systems (RMS) by halting production processes at unpredictable stages. Therefore, it is essential to manage these issues effectively to ensure smooth production in RMS. Our study concentrated on verifying two key properties: the absence of deadlocks and the absence of livelocks. We will eliminate any reconfigurable production networks (RTPMs) that exhibit these problematic behaviors. The reachability graph (RG) of a TPM is a directed graph that covers all of reachable markings and transitions among these markings. If there is some marking $M \in R$ not enabling any transition, M is said a dead marking, and system (N, M_0) is said to contain a deadlock. If there is some subset $E \subseteq R$ such that no transition leads from a marking in E to some marking not in E witnessing some progress (e.g. the initial marking), then set E is said to be a livelock, and system (N, M_0) is said to contain a livelock. The occurrence of a deadlock or livelock in a real RMS can cause severe damages.

Definition 3 (Deadlock). Let N = (P, T, F, M_0) be a Token-Based Process Model and M_f is a target marking. M ∈ R (N, M_0) is a deadlock w.r.t. M_f (deadlock for short) if ∀ t ∈ T, ¬M [t⟩ ∧ M ≠ M_f [20].

Definition 4 (Livelock). Let N = (P, T, F, M_0) be a Token-Based Process Model and M_f is a target marking. M ∈ R (N, M_0) is a livelock w.r.t. M_f (livelock for short) if ∀ M′ ∈ R (N, M), ∃ t ∈ T, M′ [t⟩ ∧ M ′ ≠ M_f [20].

A graph is strongly connected if; every node (vertex) is reachable from every other node. A strongly connected component (SCC) of a graph is a strongly connected subgraph that is maximal. An RG can be split into the live zone (LZ) and the deadlock zone (DZ). The LZ can easily be identified as the SCC containing the initial marking M_0, while the DZ can be identified as the remaining SCCs [21]. It is important to determine the exact number of states in both the DZ and LZ of the Token-Based Process Model (TPM) model. To tackle this challenge, [22] recently proposed a method that utilizes RG analysis results from a TPM model of a flexible manufacturing system (FMS) that is susceptible to deadlocks or livelocks. This method employs the Integrated Net Analyzer (INA) [21]. It takes the RG and the initial SCC as inputs and calculates the number of states in both the LZ and DZ (for a detailed algorithm, refer to [22]). The approach for verifying the RTPMs modelled in the process plan within the resource management system (RMS) is based on this methodology, as outlined in the subsequent algorithm.

Algorithm 1: Verification$_{RPN}$

Input: RPN model.

Initialisation: result_verification= True

Definitions: *nbr_sts_lz*: number of states in the LZ, *nbr_sts_dz*: the number of states in the DZ, *t_nbr_sts*: total number of states in RG, *nbr_scc*: the number of states in the first SCC

1: Compute the RG and then SCC of the given RPN model by using INA.

2: Compute the total number of states (t_nbr_sts) in RG

3: Compute the number of states in the first SCC (*nbr_scc*)

4: *nbr_sts_dz* = *t_nbr_sts* - *nbr_scc*
nbr_sts_lz = *nbr_scc*

5: if ((*nbr_sts_dz* > 0) or (*nbr_sts_lz* ≠ *t_nbr_sts*)) then result_verification= False

Output: result_verification

End of Algorithm.

Source: Authors, (2026).

VII. NS̃GA-III\LEPN ALGORITHM

The series of non-dominated sorting based genetic algorithms (NSGA-series) has clearly shown their niche in solving multi- and many objective optimization problems. Of them, NSGA-III [23], [24] was designed to solve problems having three or more objectives efficiently. NSGA-III serves as the foundation for several many-objective optimization methods, with a set of extensions proposed U-NSGA-III [25], EliteNSGA-III [26], DC-NSGA-III [27], R-NSGA-III [28], NSGA-III\NDS [29]. To generate optimal process plans for sustainable manufacturing criteria, we attempt to use last extension of NSGA-III; the NSGA-III without non-dominated sorting (NDS) procedure namely, NS̃GA-III or NSGA-III\NDS. The basic framework of NS̃GA-III algorithm is similar to NSGA-III with significant changes in the way (i) domination check is executed, (ii) the survival selection operator is modified, and (iii) a few other minor changes are adopted resulting from the change in domination check procedure.

VII.1 PROPOSED ALGORITHM

Algorithm 2 describes the proposed algorithm.

Algorithm 2: NS̃GA-III\ LEPN for process plan generation in SRMS.

Input: Operations precedence DAG

OP= Set of operations, M= Set of available machines, CT= Set of Configurations and tools that each machine offers
Predefined set of reference directions W_r
Set of Parameters /* detailed in section 5.1 */
St = ∅, no of selections remaining (nr) = N

/* Create a set of mappings, where each mapping corresponds to a distinct individual or solution. */
1 M_p= Search for Feasible Mappings (OP, M, CT)
/* Transform each mapping into its LEPN representation */
2 Individuals= chromosome encoding (M_p, DAG)
/* Analyse the Net and Determine Transition Firing Order*/
3 Parent Population P_t= Evaluate Each Individual (Individuals)
/* Offspring population generation */
4 Q_t = Generate child Population from parent Population (Pt) using: $Crossover_{RPN}$, $Mutation_{RPN}$, $Verification_{RPN}$
5 $R_t = P_t \cup Q_t$
6 I = find-non-dominated (R_t)
/* Finding ideal point and nadir point for normalization of objective space */
7 ideal, nadir = hyperplane-boundary-estimation (R_t, I)
/* Association each s in R_t with a reference direction */
8 [π(s), d_2(s)] = associate-to-niches (R_t, W_r, ideal, nadir)
/* π(s) = closest reference direction, d_2(s) = perpendicular distance between π(s) and s */
9 d(s) = pbi-decomposition (R_t, W_r, θ=5, ideal, nadir)
/* d(s) = d_1+θd_2 distance between π(s) and s, d1 distance along reference direction ideal point */
10 Set attribute to each population member non-dominated ND=1 and dominated ND=0
11 Set attributes each population member R_t (CV, ND, d(s))
/*classifying population members */
12 Class 1 (C1): R_t (CV = 0 ∩ ND = 1)
13 Class2 (C2): R_t (CV=0 ∩ ND=0)
14 Class3 (C3): R_t (CV>0)
/*Class 1 selection */
15 if (|C1|≤ N) then $S_t = S_t \cup C1$; $n_r = N - |S_t|$
16 else S_t, n_r = class-survivor-selection (C1, S_t, n_r, W_r, π(s), d(s)); $P_{t+1} = S_t$, break;
/*Class 2 selection */
17 if n_r> 0 then S_t, n_r = class-survivor-selection (C2, S_t, n_r, W_r, π(s), d(s));
18 else $P_{t+1} = S_t$; break;
/*Class 3 selection for constrained problems */
19 if n_r> 0 then S_t = tournament-selection (C3);
20 else P_{t+1}=St; break ;

Result: P_{t+1}

Source: Authors, (2026).

VII.2 ALGORITHM DESCRIPTION

The proposed algorithm begins by creating a set of mappings (Step 1), where each mapping represents a distinct individual or solution, serving as a candidate for optimization. These mappings are then transformed into their corresponding LEPN (Labeled Evolutionary Petri Net) representations (Step 2). Following this transformation, the algorithm analyzes the Petri Net and determines the transition firing order (Step 3). This order specifies the sequence in which transitions are activated within the net and is critical for managing the flow of operations, ultimately determining how solutions evolve throughout the optimization process. In Steps 4 and 5, the algorithm generates an offspring population through genetic operations, specifically crossover and mutation. These operations introduce variability into the population, creating new individuals that explore different regions of the solution space. Subsequently, the population is normalized by identifying the ideal point and the nadir point in Steps 6 and 7. These reference points are essential for scaling the objective values across the population, ensuring that comparisons between solutions are made on a consistent and comparable basis. In Step 8, each individual solution ss is associated with a reference direction. Specifically, the closest reference direction π(s) is identified for each solution, and the perpendicular distance d2(s) between the individual solution ss and the reference direction π(s) is computed. This relationship is crucial for evaluating each solution's position relative to the reference direction, providing insight into the distribution of solutions in the objective space. In Step 9, the algorithm applies the PBI (Penalized Boundary Intersection) decomposition. The distance d(s) between an individual solution ss and its closest reference direction is calculated as the sum of two components: the distance along the reference direction d_1(s), and the perpendicular distance d_2(s). The subsequent steps of the algorithm focus on classifying the population members.

VIII. EXPERIMENTATION VALIDATION

The proposed approach is experimentally validated through the use of Petri Net Markup Language (PNML[i]) and the TiNA[ii] Tools software. Figure 4 depicts the steps of validation. TiNA Tools is employed to validate the Petri net represented in a PNML file, confirming its correctness. The tool can analyse all firing sequences of a Petri net related to a process plan problem. Each sequence of transitions, starting from the initial marking and leading to the final marking, corresponds to a feasible process plan. The transitions from the initial state S_0 to the final state S_f represent an ordered execution of operations and a structured utilization of triplets (machine-configuration-tool). This sequence is then saved in a stepper file, which is used to label the Petri net and assess the fitness of each individual.

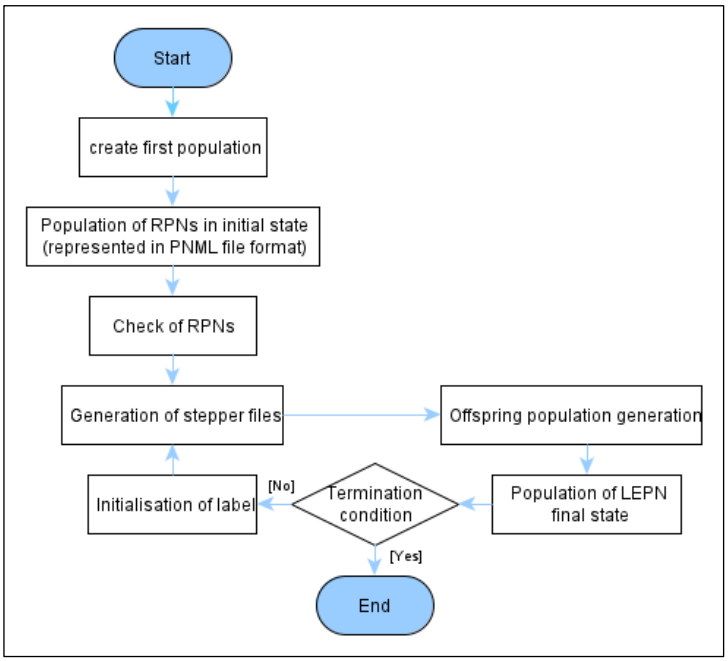

Figure 4: Steps of experimental validation.
Source: Authors, (2026).

The justification for using the PNML format rests on three key aspects: readability, universality, and mutuality. Consequently, the format must capture the fundamental principles and common notations of Petri nets, thereby facilitating the manipulation of labelled evolutionary Petri nets (LEPNs), particularly during the labelling phase. This design also supports the application of both crossover and mutation operators. The experimental results were conducted using an Intel Core i5 processor running at 1.8 GHz and 8 GB of RAM. The meta-heuristic algorithms were implemented using Python, specifically utilizing the **pymoo**[iii] framework for multi-objective optimization. The parameters used in this study are a population size of 100 individuals, a selection rate of 50% per iteration, a crossover rate of 50%, a mutation rate of 5%, and a maximum of 1000 iterations. Figures 5 and 6 present key results from our application in the case of 100 operations and 20 reconfigurable machines. Figure 5 shows the distribution of the initial population in the first iteration, while Figure 6 depicts the solution distribution after the algorithm has converged. Figures 7-10 illustrate the algorithm's convergence regarding the objectives cost, time, LHW and GHG over a single run of 1000 iterations.

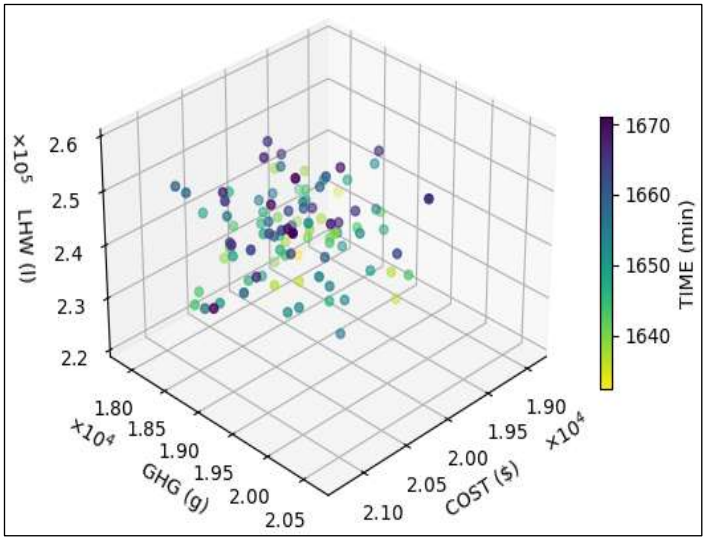

Figure 5: Distribution of the Initial Population.
Source: Authors, (2026).

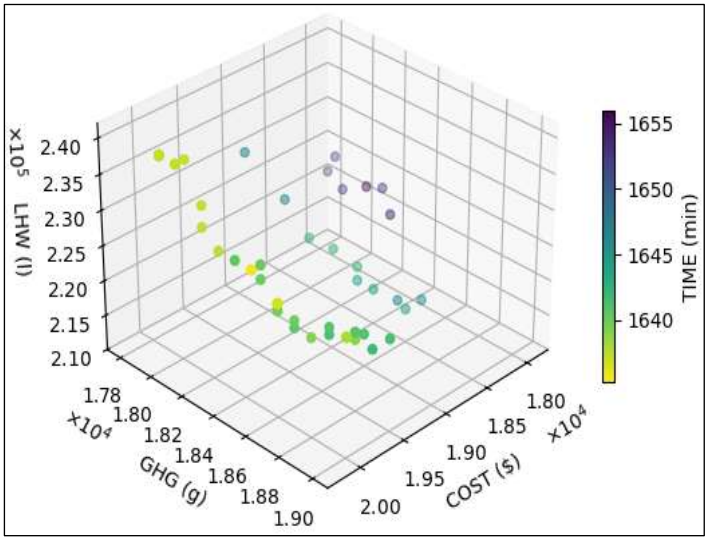

Figure 6: Distribution of Solutions.
Source: Authors, (2026).

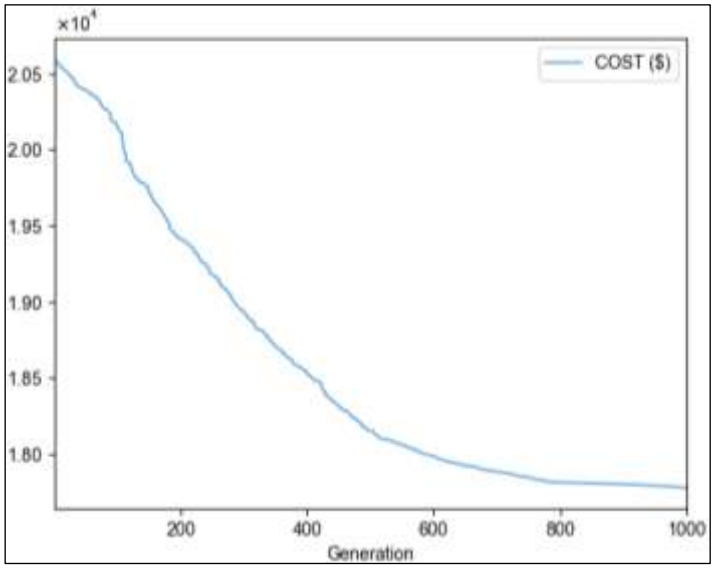

Figure 7: Convergence of the algorithm (Cost objective).
Source: Authors, (2026).

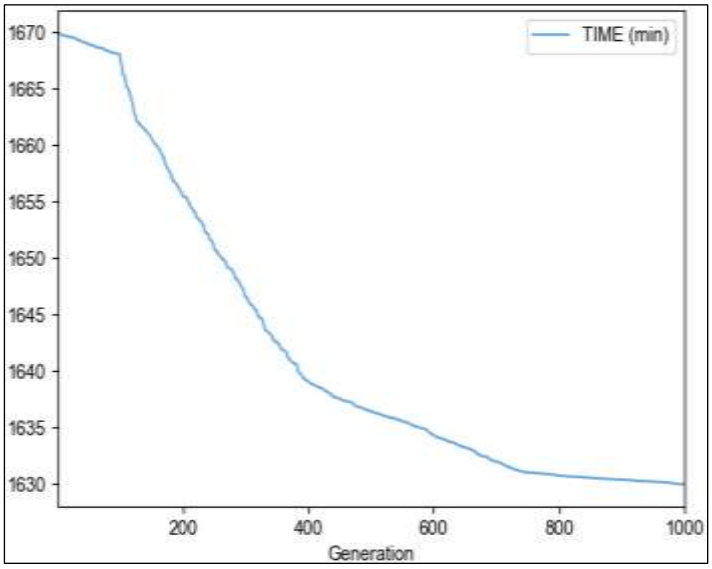

Figure 8: Convergence of the algorithm (Time objective).
Source: Authors, (2026).

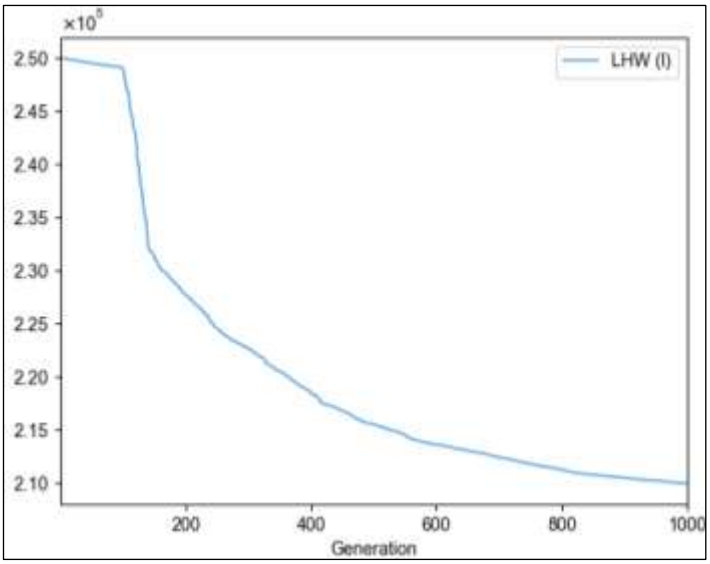

Figure 9: Convergence of the algorithm (LHW objective).
Source: Authors, (2026).

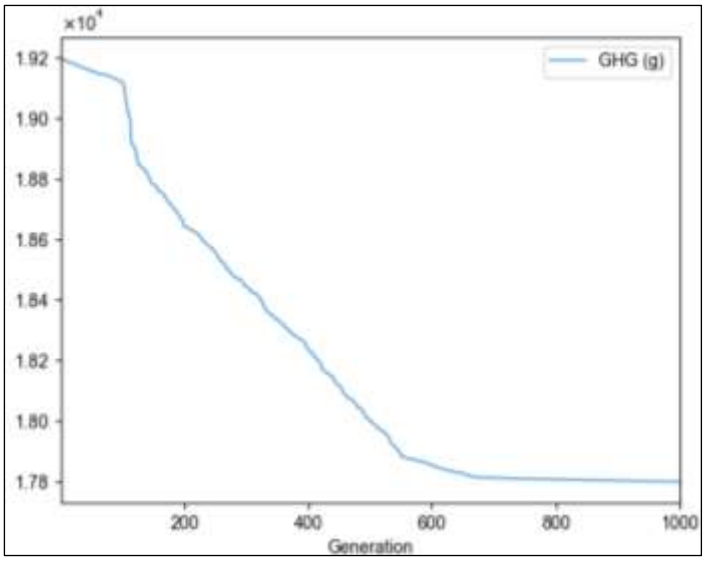

Figure 10: Convergence of the algorithm (GHG objective).
Source: Authors, (2026).

VIII.1 INFLUENCE OF GENETIC OPERATOR RATES ON ALGORITHM CONVERGENCE

VIII.1.1 Influence of the crossover rate

In our experiment, we evaluated the performance of our algorithm by varying the crossover rate. The results showed that the best outcomes for cost, time, GHG emissions, and liquid waste (LHW) were achieved when the crossover rate was set to 0.5. When the crossover rate is increased to 0.7, the algorithm performs well for the **time** and **liquid waste (LWH)** objectives compared to **cost** and **GHG emissions**. This suggests that a higher crossover rate promotes faster convergence and prioritizes minimizing time and waste production. Conversely, when the crossover rate is decreased to 0.3, the algorithm performs better for the **cost** and **GHG emissions** objectives, indicating that a lower crossover rate facilitates a more thorough exploration of the solution space, leading to more balanced solutions with respect to economic and environmental considerations, albeit at the cost of slightly higher values for time and LWH. These findings highlight the trade-offs associated with adjusting the crossover rate, where a higher rate may improve certain objectives at the expense of others, and vice versa. Figures 11-14 illustrate the effects of varying the crossover rate within the range of [0.3, 0.7] on cost, time, GHG emissions, and liquid waste (LHW).

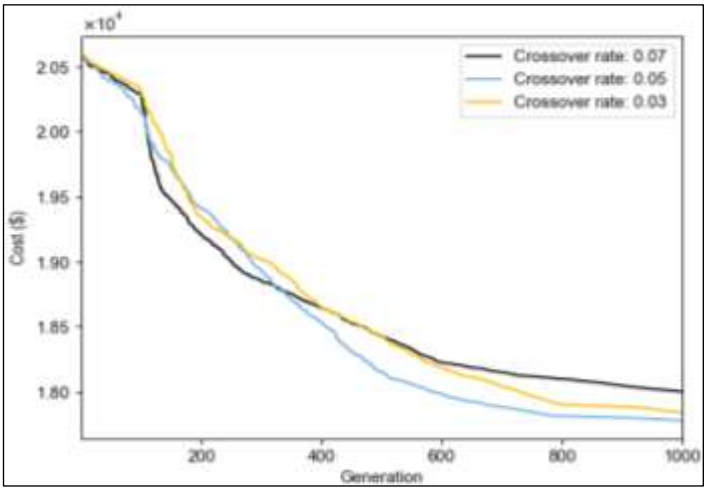

Figure 11: Influence of crossover rate Cost objective.
Source: Authors, (2026).

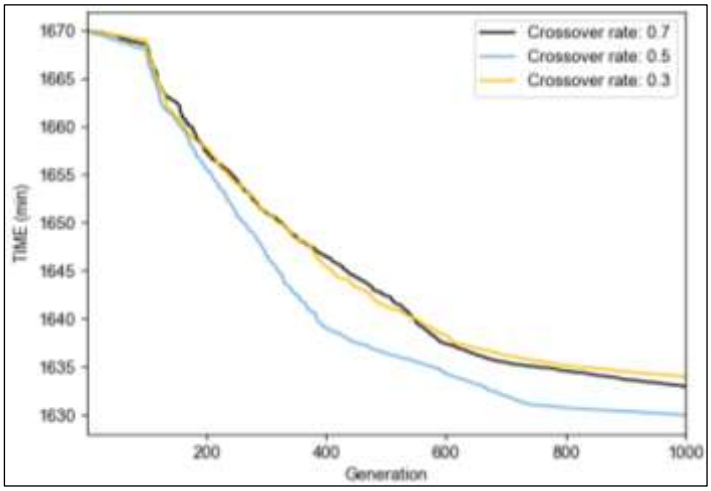

Figure 12: Influence of crossover rate Time objective.
Source: Authors, (2026).

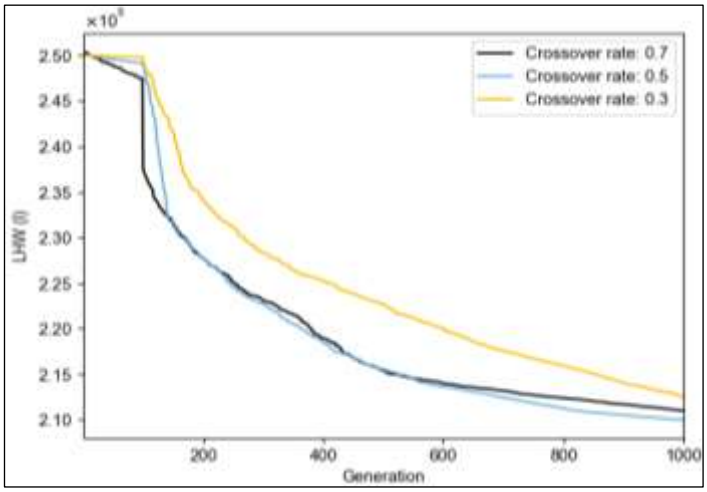

Figure 13: Influence of crossover rate liquid waste (LHW) objective.
Source: Authors, (2026).

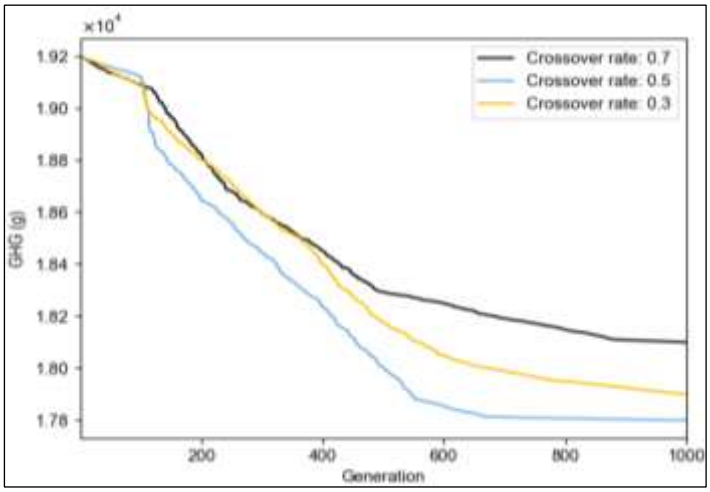

Figure 14: Influence of crossover rate GHG emissions objective.
Source: Authors, (2026).

VIII.1.2 Influence of the mutation rate

We tested our algorithm with different mutation rates and observed the following trends. When the mutation rate is set to 0.05, the algorithm yields the best results for all four objectives: **cost, time, liquid waste (LWH)**, and **GHG emissions**. However, when the mutation rate is increased to 0.07, the algorithm performs well for the objectives of **time, liquid waste (LWH)**, and **GHG emissions**, but not for **cost**. Conversely, when the mutation rate is decreased to 0.03, the algorithm performs better for **cost** but at the expense of the other objectives, particularly time and liquid waste.

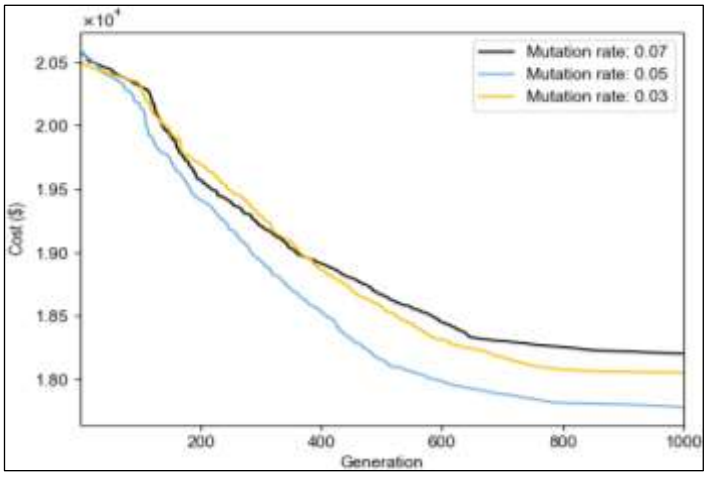

Figure 15: Influence of mutation rate Cost objective.
Source: Authors, (2026).

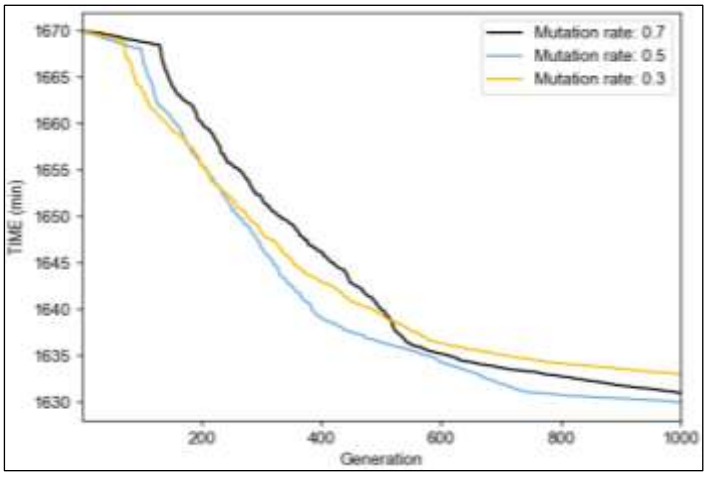

Figure 16: Influence of mutation rate Time objective.
Source: Authors, (2026).

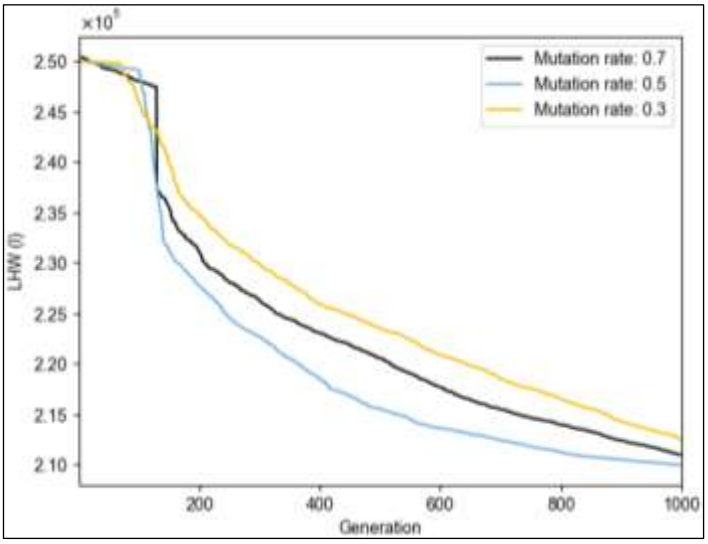

Figure 17: Influence of mutation rate liquid waste (LHW) objective.
Source: Authors, (2026).

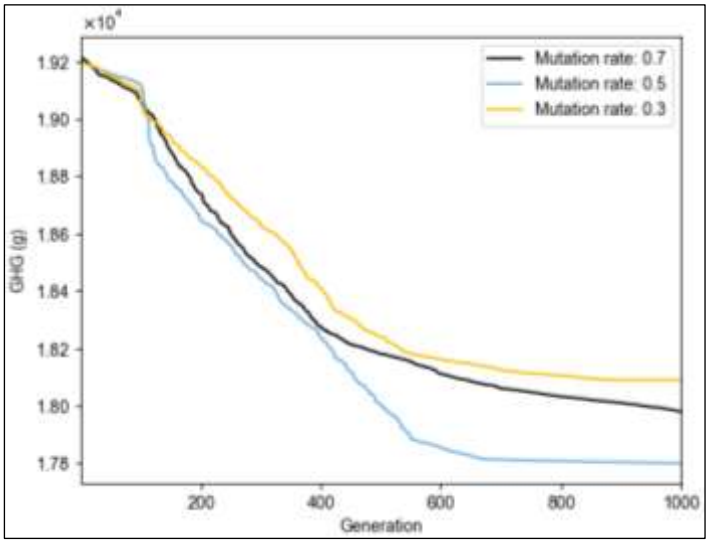

Figure 18: Influence of mutation rate GHG emissions objective.
Source: Authors, (2026).

VIII.2 COMPARISON WITH OTHER ALGORITHMS

In the context of a single-unit scenario, an instance is defined by the number of requested operations and the number of available reconfigurable machines. Figures 19 and 20 summarize the results obtained from the experiments for each instance, evaluated using the metaheuristics NSGA-II, NSGA-III, and the proposed approach NS̃GA-III\LEPN. Figure 19 presents the CPU computation time (in seconds). Compared to NSGA-II and NSGA-III, our algorithm demonstrates a computation time that is between the two for small-sized instances, while it outperforms both for medium and large-sized instances. Figure 20 shows the cardinality of the Pareto fronts, representing the number of Pareto-optimal process plans. For small and medium-sized instances, our algorithm exhibits cardinality between that of NSGA-II and NSGA-III, whereas, for large-sized instances, it achieves a higher cardinality than both.

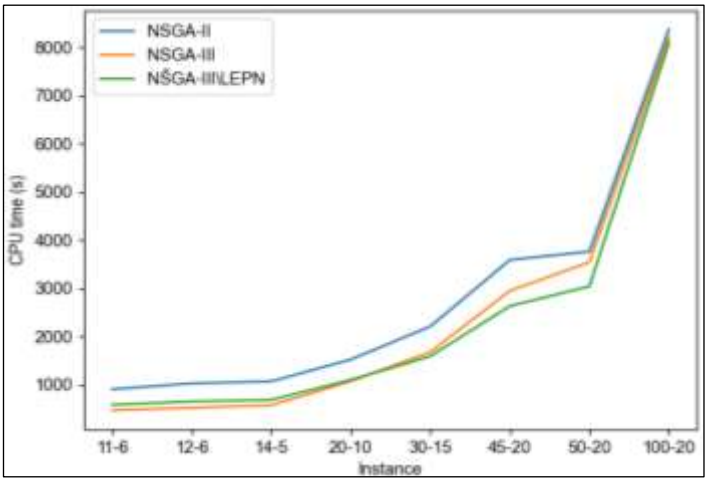

Figure 19: CPU time of the metaheuristic Methods.
Source: Authors, (2026).

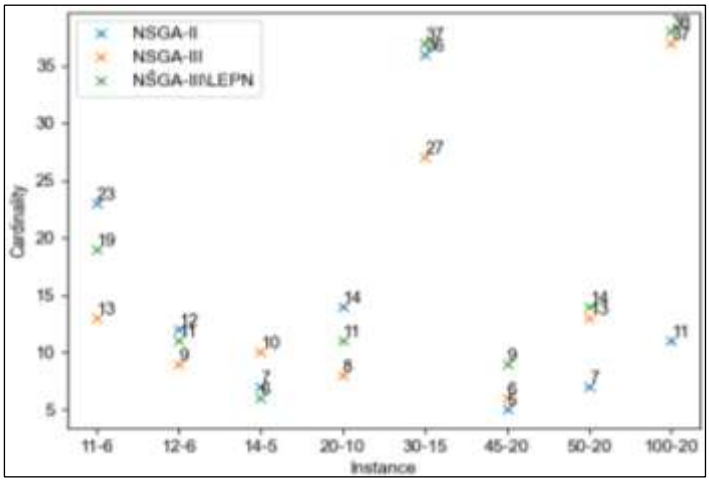

Figure 20: Cardinality of pareto fronts generated by metaheuristic Methods.
Source: Authors, (2026).

IX. CONCLUSION

In conclusion, this study presents an environmentally-oriented multi-objective process planning approach for sustainable reconfigurable manufacturing systems (SRMS), with a focus on minimizing four key objectives: total production cost, production time, greenhouse gas emissions from energy consumption, and hazardous liquid waste. We integrated Petri Nets with a Genetic Algorithm (GA) to derive Pareto-optimal solutions, using a variant of the non-dominated sorting genetic algorithm, NSGA-III, referred to as NS̃GA-III. The experimental results demonstrate the effectiveness of our proposed approach, emphasizing the advantages of incorporating Petri Nets in this context. Notably, the NS̃GA-II\LEPN algorithm outperforms both NSGA-II and NSGA-III, particularly in terms of CPU time for medium and large-sized instances, as well as in the cardinality of Pareto fronts for larger instances. In future work, we plan to extend the proposed approach by testing it on additional case studies to further assess its robustness and performance. Furthermore, we aim to introduce additional objectives, such as maximizing the amount of product produced per unit of energy consumed or enhancing system reliability by maximizing the Mean Time Between Failures (MTBF). This will enable us to evaluate the ability of our approach to balance between the objectives that need to be minimized and those that should be maximized, ultimately improving the overall performance and sustainability of the system.

X. REFERENCES

[1] A. Dahmani, L. Benyoucef, and J.-M. Mercantini, "Toward Sustainable Reconfigurable Manufacturing Systems (SRMS): Past, Present, and Future," Procedia Computer Science, vol. 200, pp. 1605–1614, 2022, doi: https://doi.org/10.1016/j.procs.2022.01.361.

[2] J. Plucar, Ondrej Grunt, Pandian Vasant, and I. Zelinka, "Modelling Business Processes Using Evolutionary Generated Petri Nets," Jan. 2017, doi: https://doi.org/10.4108/eai.27-2-2017.152272.

[3] N. Brahimi, A. Dolgui, E. Gurevsky, and A. R. Yelles-Chaouche, "A literature review of optimization problems for reconfigurable manufacturing systems," IFAC-PapersOnLine, vol. 52, no. 13, pp. 433–438, 2019, doi: https://doi.org/10.1016/j.ifacol.2019.11.097.

[4] B. Assou, I. Khettabi, and L. Benyoucef, "Operators Health and Safety Consideration in Sustainable Multi-objective Process and Production Planning for Reconfigurable Manufacturing System (RMS)," 2022 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 516–520, Dec. 2022, doi: https://doi.org/10.1109/ieem55944.2022.9989727.

[5] M. A. Yazdani, A. Khezri, and L. Benyoucef, "Process and production planning for sustainable reconfigurable manufacturing systems (SRMSs): multi-objective exact and heuristic-based approaches," The International Journal of Advanced Manufacturing Technology, Jan. 2022, doi: https://doi.org/10.1007/s00170-021-08409-0.

[6] M. A. Yazdani, A. Khezri, and L. Benyoucef, "A Linear Multi-Objective Optimization Model for Process and Production Planning Generation in a Sustainable Reconfigurable Environment," IFAC-PapersOnLine, vol. 54, no. 1, pp. 689–695, Jan. 2021, doi: https://doi.org/10.1016/j.ifacol.2021.08.181.

[7] M. A. Yazdani, L. Benyoucef, A. Khezri, and A. Siadat, "Multi-objective Process and Production Planning Integration in Reconfigurable Manufacturing Environment: Augmented E-constraint Based Approach," 13th International Conference on MOdeling, Optimization and SIMulation (MOSIM 2020):"New Advances and Challenges for Sustainable and Smart Industries", pp. 552-557, Nov. 2020.

[8] Paolo Gianessi, A. Cerqueus, D. Lamy, and X. Delorme, "Using Reconfigurable Manufacturing Systems to minimize energy cost: a two-phase algorithm," IFAC-PapersOnLine, vol. 54, no. 1, pp. 379–384, Jan. 2021, doi: https://doi.org/10.1016/j.ifacol.2021.08.042.

[9] I. Khettabi, L. Benyoucef, and M.-A. Boutiche, "A Comparative Study of Three NSGA Versions for the Multi-Objective Sustainable Process Plans Optimization in RMS," IFAC-PapersOnLine, vol. 55, no. 10, pp. 785–790, 2022, doi: https://doi.org/10.1016/j.ifacol.2022.09.505.

[10] I. Khettabi, L. Benyoucef, and M. A. Boutiche, "Sustainable Multi-objective Process Plan Generation in RMS: Dynamic NSGA-II vs New Dynamic NSGA-II," 2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Dec. 2021, doi: https://doi.org/10.1109/ieem50564.2021.9672869.

[11] I. Khettabi, L. Benyoucef, and M. A. Boutiche, "Sustainable reconfigurable manufacturing system design using adapted multi-objective evolutionary-based approaches," The International Journal of Advanced Manufacturing Technology, vol. 115, no. 11–12, pp. 3741–3759, Jun. 2021, doi: https://doi.org/10.1007/s00170-021-07337-3.

[12] I. Khettabi, M. A. Boutiche, and L. Benyoucef, "NSGA-II vs NSGA-III for the Sustainable Multi-Objective Process Plan Generation in a Reconfigurable Manufacturing Environment," IFAC-PapersOnLine, vol. 54, no. 1, pp. 683–688, 2021, doi: https://doi.org/10.1016/j.ifacol.2021.08.180.

[13] F. A. Touzout and L. Benyoucef, "Multi-objective sustainable process plan generation in a reconfigurable manufacturing environment: exact and adapted evolutionary approaches," International Journal of Production Research, vol. 57, no. 8, pp. 2531–2547, Sep. 2018, doi: https://doi.org/10.1080/00207543.2018.1522006.

[14] F. A. Touzout and Lyes Benyoucef, "Multi-objective multi-unit process plan generation in a reconfigurable manufacturing environment: a comparative study of three hybrid metaheuristics," International Journal of Production Research, vol. 57, no. 24, pp. 7520–7535, Jul. 2019, doi: https://doi.org/10.1080/00207543.2019.1635277.

[15] A. Khezri, H. H. Benderbal, and L. Benyoucef, "Towards a sustainable reconfigurable manufacturing system (SRMS): multi-objective based approaches for process plan generation problem," International Journal of Production Research, vol. 59, no. 15, pp. 4533–4558, May 2020, doi: https://doi.org/10.1080/00207543.2020.1766719.

[16] A. Khezri, H. Haddou Benderbal, and Lyes Benyoucef, "Sustainable Multi-objective Process Plan Generation in RMS Through Modelling Energy Consumption," Springer series in advanced manufacturing, pp. 161–177, Oct. 2019, doi: https://doi.org/10.1007/978-3-030-28782-5_8.

[17] S. Gao, J. Daaboul, and J. Le Duigou, "Process Planning, Scheduling, and Layout Optimization for Multi-Unit Mass-Customized Products in Sustainable Reconfigurable Manufacturing System," Sustainability, vol. 13, no. 23, p. 13323, Dec. 2021, doi: https://doi.org/10.3390/su132313323.

[18] M. Femmam, O. Kazar, L. Kahloul, and M. E. K. Fareh, "Labelled evolutionary Petri nets/genetic algorithm based approach for workflow scheduling in cloud computing," International Journal of Grid and Utility Computing, vol. 9, no. 2, p. 157, 2018, doi: https://doi.org/10.1504/ijguc.2018.091721.

[19] M. S. Nobile, D. Besozzi, Paolo Cazzaniga, and G. Mauri, "The foundation of Evolutionary Petri Nets," Munich Personal RePEc Archive (Ludwig Maximilian University of Munich), vol. 988, no. 3, pp. 60–74, Jan. 2013.

[20] G. Liu, K. Zhang, and C. Jiang, "Deciding the Deadlock and Livelock in a Petri Net with a Target Marking Based on Its Basic Unfolding," Lecture notes in computer science, pp. 98–105, Jan. 2016, doi: https://doi.org/10.1007/978-3-319-49583-5_7.

[21] M. Uzam et al., "On Deadlock/Livelock Studies Based on Reachability Graph of Petri Nets by Using TINA," IEEE Access, vol. 12, pp. 135506–135534, 2024, doi: https://doi.org/10.1109/access.2024.3461168.

[22] Gökhan GELEN, "Computation of the number of legal states for petri net-based deadlock prevention problems," Sigma Journal of Engineering and Natural Sciences – Sigma Mühendislik ve Fen Bilimleri Dergisi, Jan. 2023, doi: https://doi.org/10.14744/sigma.2023.00056.

[23] K. Deb and H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints," IEEE Transactions on Evolutionary Computation, vol. 18, no. 4, pp. 577–601, Aug. 2014, doi: https://doi.org/10.1109/tevc.2013.2281535.

[24] K. Deb and H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints," IEEE Transactions on Evolutionary Computation, vol. 18, no. 4, pp. 577–601, Aug. 2014, doi: https://doi.org/10.1109/tevc.2013.2281535.

[25] H. Seada and K. Deb, "A Unified Evolutionary Optimization Procedure for Single, Multiple, and Many Objectives," IEEE Transactions on Evolutionary Computation, vol. 20, no. 3, pp. 358–369, Jun. 2016, doi: https://doi.org/10.1109/tevc.2015.2459718.

[26] A. Ibrahim, Shahryar Rahnamayan, Miguel Vargas Martín, and K. Deb, "EliteNSGA-III: An improved evolutionary many-objective optimization algorithm," Jul. 2016, doi: https://doi.org/10.1109/cec.2016.7743895.

[27] H. Seada, M. Abouhawwash, and K. Deb, "Towards a Better Balance of Diversity and Convergence in NSGA-III: First Results," Lecture notes in computer science, pp. 545–559, Jan. 2017, doi: https://doi.org/10.1007/978-3-319-54157-0_37.

[28] Y. Vesikar, K. Deb, and J. Blank, "Reference Point Based NSGA-III for Preferred Solutions," Nov. 2018, doi: https://doi.org/10.1109/ssci.2018.8628819.

[29] B. Santoshkumar, K. Deb, and L. Chen, "Eliminating Non-dominated Sorting from NSGA-III," in Lecture notes in computer science, 2023, pp. 71–85. doi: 10.1007/978-3-031-27250-9_6.

[i] https://www.pnml.org/
[ii] https://projects.laas.fr/tina/home.php
[iii] https://pymoo.org/