People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Mohamed Khider - BISKRA
Faculty of Exact Sciences, Natural and Life Sciences
Computer Science Department

Order Number:
Serial Number:

Thesis

In Candidacy for the Degree of

Doctor 3rd Cycle in Computer Science
Option: Artificial Intelligence

Title

Formal Modeling and Verification of Reconfigurable Systems

By:
Houimli Manel

Defended on 02 November 2025, before a jury composed of:

Chairman:	Prof. Bennoui Hammadi, Professor, Biskra University, Algeria.
Supervisor:	Prof. Kahloul Laid, Professor, Biskra University, Algeria.
Co-Supervisor:	Prof. Khalgui Mohamed, Professor, GIU-Berlin University, Germany.
Examiner:	Prof. Chaoui Allaoua, Professor, Constantine 2 University, Algeria.
Examiner:	Dr. Tigane Samir, MCA, Biskra University, Algeria.

ملخص

تتطلب تعقيدات الأنظمة الحديثة وطبيعتها الديناميكية منهجيات متقدمة لضمان تكيفها وكفاءتها. وتعد الأنظمة القابلة لإعادة التكوين، التي تتميز بقدرتها على التكيف بسرعة مع المتطلبات المتغيرة، من أكبر التحديات في النمذجة والتحقق والتحسين. وقد ركزت العديد من جهود البحث على استخدام الأساليب الرسمية لمواجهة هذه التحديات، بهدف تعزيز موثوقية وأداء هذه الأنظمة في تطبيقات متنوعة.

لقد كانت الأساليب الرسمية التقليدية، مثل Automata و شبكات بيتري الكلاسيكية (PNs)، فعالة في النمذجة والتحقق، لكنها تعاني عند تطبيقها على تعقيدات الأنظمة القابلة لإعادة التكوين. فالتغييرات الهيكلية المتكررة تتطلب تقنيات تحقق تأخذ في الاعتبار الانتقالات الديناميكية. وللتعامل مع هذه المشكلة، قام الباحثون بتوسيع الأساليب الرسمية لتشمل القابلية لإعادة التكوين، مما أدى إلى تطوير تقنيات جديدة للنمذجة والتحقق (مثل توسعات model checking، Automata الاحتمالية، التوسعات الخاصة بشبكات بيتري المعتمدة على القابلية لإعادة التكوين، والمنطق الاحتمالي PCTL) والتي تم تصميمها خصيصًا لهذه الأنظمة. وفي الوقت نفسه، تم استكشاف تقنيات التحسين بشكل واسع بعيدًا عن التحقق الرسمي لتحديد التكوينات المثلى في مراحل إعادة التكوين المختلفة. فعلى سبيل المثال، تعمل الطرق مثل الخوارزميات الجينية والبحث الاسترشادي على تحسين تخصيص الموارد والأداء، لكنها تفتقر إلى الضمانات الرسمية بالنسبة للصحة والسلامة. وتسلط هذه الفجوة الضوء على الحاجة إلى منهجيات تجمع بين التحسين والتحقق الرسمي لضمان الكفاءة والموثوقية في الأنظمة القابلة لإعادة التكوين.

في هذه الأطروحة، نقدم نهجين مبتكرين للنمذجة والتحقق وتحسين الأنظمة القابلة لإعادة التكوين. تساهم المساهمة الأولى بتقديم إطار قوي لنمذجة بروتوكولات الاتصال في طبقة MAC لشبكات الاستشعار اللاسلكية المتنقلة (MWSNs) وبروتوكول MQTT المستخدم في تطبيقات إنترنت للأشياء(IoT). من خلال استخدام Automata الاحتمالية للنمذجة والمنطق الاحتمالي (PCTL) لتحديد الخصائص، نستخدم UPPAAL SMC للتحقق من النماذج بشكل دقيق. لا تحسن هذه الطريقة موثوقية هذه البروتوكولات فحسب، بل توفر أيضًا طريقة منهجية للتحقق من الخصائص الحرجة، مما يعزز الثقة في تطبيقات الإنترنت للأشياء.

المساهمة الثانية والرئيسية تقترح صيغة جديدة تسمى شبكات الكائنات القابلة لإعادة التكوين الجينية(Gen-RONs)، والتي تمكّن من النمذجة والتحقق من الهياكل الديناميكية للأنظمة القابلة لإعادة التكوين بينما تعمل على تحسينها. من خلال دمج النمذجة الرسمية والتحقق والتحسين في إطار متكامل، نقوم بتحديد عمليات الطفرة والتقاطع للخوارزميات الجينية من خلال تفعيل الانتقالات الخاصة بالقواعد في شبكات الكائنات القابلة لإعادة التكوين. تقدم هذه الطريقة المبتكرة تحسنًا كبيرًا في قدرات الأنظمة القابلة لإعادة التكوين، حيث توفر حلاً شاملاً يعالج تعقيدات القابلية لإعادة التكوين الديناميكي جنبًا إلى جنب مع استراتيجيات التحسين الفعالة.

الكلمات المفتاحية: الأنظمة القابلة لإعادة التكوين، النمذجة الرسمية، التحقق الرسمي، التحسين، الخوارزميات الوراثية، نظرية تحويلات الرسوم البيانية.

Abstract

The increasing complexity and dynamic nature of modern systems require advanced approaches to ensure their adaptability and efficiency. Reconfigurable systems, characterized by their ability to rapidly adjust to changing requirements, present significant challenges in modeling, verification, and optimization. Numerous research efforts have focused on employing formal methods to tackle these challenges, aiming to enhance the reliability and performance of these systems in diverse applications.

Traditional formal methods, such as automata and classical Petri nets (PNs), have been effective for modeling and verification but struggle with the complexity of reconfigurable systems. Frequent structural changes require verification techniques that account for dynamic transitions. To address this, research has extended formal methods to incorporate reconfigurability, leading to new modeling and verification techniques (such as model checking extensions, probabilistic automata, reconfigurability-based Petri nets extensions and Probabilistic Computation Tree Logic) tailored to these systems. Meanwhile, optimization techniques have been widely explored independently of formal verification to determine optimal configurations at different reconfiguration stages. Methods like genetic algorithms and heuristic search optimize resource allocation and performance but lack formal guarantees of correctness and safety. This gap highlights the need for approaches that combine optimization and formal verification to ensure both efficiency and reliability in reconfigurable systems.

In this thesis, we present two innovative approaches for modeling, verifying, and optimizing reconfigurable systems. The first contribution introduces a robust framework for modeling communication protocols at the Medium Access Control (MAC) layer of mobile wireless sensor networks (MWSNs) and the MQTT protocol used in IoT applications. By utilizing probabilistic automata for modeling and PCTL for property specification, we employ UPPAAL SMC for rigorous model checking. This approach not only enhances the reliability of these protocols but also provides a systematic method for verifying critical properties, ultimately fostering trust in IoT applications.

The second and major contribution proposes a novel formalism called Genetic Reconfigurable Object Nets (Gen-RONs), which enables the modeling and verification of the dynamic structures of reconfigurable systems while optimizing them. By integrating formal modeling, verification, and optimization into a cohesive framework, we define mutation and crossover operations of genetic algorithms through the formal firing of rule transitions in Reconfigurable Object Nets. This innovative approach significantly advances the capabilities of reconfigurable systems, offering a comprehensive solution that addresses the complexities of dynamic reconfigurability alongside effective optimization strategies.

Key words: Reconfigurable Systems, Formal Modeling, Formal Verification, Optimization, Genetic Algorithms, Graph Transformation Theory.

Résumé

La complexité croissante et la nature dynamique des systèmes modernes exigent des approches avancées pour garantir leur adaptabilité et leur efficacité. Les systèmes reconfigurables, caractérisés par leur capacité à s'ajuster rapidement aux exigences changeantes, posent des défis importants en matière de modélisation, de vérification et d'optimisation. De nombreux travaux de recherche se sont concentrés sur l'utilisation des méthodes formelles pour relever ces défis, dans le but d'améliorer la fiabilité et les performances de ces systèmes dans diverses applications.

Les méthodes formelles traditionnelles, telles que les automates et les réseaux de Petri classiques (PNs), ont été efficaces pour la modélisation et la vérification, mais rencontrent des difficultés face à la complexité des systèmes reconfigurables. Les changements structurels fréquents nécessitent des techniques de vérification qui prennent en compte les transitions dynamiques. Pour répondre à cela, des recherches ont élargi les méthodes formelles pour intégrer la reconfigurabilité, ce qui a conduit à de nouvelles techniques de modélisation et de vérification (telles que les extensions de la vérification de modèles, les automates probabilistes, les extensions des réseaux de Petri basés sur la reconfigurabilité et la logique probabiliste des arbres de calcul) adaptées à ces systèmes. Parallèlement, des techniques d'optimisation ont été largement explorées indépendamment de la vérification formelle pour déterminer les configurations optimales à différents stades de reconfiguration. Des méthodes comme les algorithmes génétiques et la recherche heuristique optimisent l'allocation des ressources et les performances, mais manquent de garanties formelles en matière de correction et de sécurité. Cette lacune met en évidence la nécessité d'approches combinant optimisation et vérification formelle pour assurer à la fois l'efficacité et la fiabilité des systèmes reconfigurables.

Dans cette thèse, nous présentons deux approches innovantes pour modéliser, vérifier et optimiser les systèmes reconfigurables. La première contribution introduit un cadre robuste pour la modélisation des protocoles de communication au niveau de la couche MAC des réseaux de capteurs sans fil mobiles (MWSNs) et du protocole MQTT utilisé dans les applications IoT. En utilisant les automates probabilistes pour la modélisation et la logique des arbres de calcul probabilistes (PCTL) pour la spécification des propriétés, nous utilisons UPPAAL SMC pour une vérification rigoureuse des modèles. Cette approche améliore non seulement la fiabilité de ces protocoles, mais elle fournit également une méthode systématique pour vérifier les propriétés critiques, favorisant ainsi la confiance dans les applications IoT.

La deuxième et principale contribution propose un formalisme novateur appelé Réseaux d'Objets Reconfigurables Génétiques (Gen-RONs), qui permet de modéliser et de vérifier les structures dynamiques des systèmes reconfigurables tout en les optimisant. En intégrant la modélisation formelle, la vérification et l'optimisation dans un cadre cohérent, nous définissons les opérations de mutation et de croisement des algorithmes génétiques par la transition formelle des règles dans les Réseaux d'Objets Reconfigurables. Cette approche innovante améliore de manière significative les capacités des systèmes reconfigurables, offrant une solution complète qui aborde les complexités de la reconfigurabilité dynamique tout en offrant des stratégies d’optimisation efficaces.

Mots-clés : Systèmes Reconfigurables, Modélisation Formelle, Vérification Formelle, Optimisation, Algorithmes Génétiques, Théorie de la Transformation de Graphes.

Dedication

This thesis is dedicated to ...

To my beloved father, whose memory and wisdom continue to inspire me every day.

To my dear mother, whose tireless love and sacrifices have been my greatest source of strength.

To my husband, whose unwavering support and encouragement have carried me through this journey.

To my children, Adem, Taha, and Noha, whose joy and love give me purpose every day.

To my siblings, whose love and support mean the world to me.

And to everyone who has helped and supported me along the way, your kindness has made this possible.

Acknowledgements

It is with profound gratitude that I acknowledge the invaluable support and guidance I have received throughout this journey. I extend my deepest thanks to my supervisor, Pr. Kahloul Laid, whose unwavering dedication and expertise have been essential to the completion of this work. His readiness to assist and his insightful advice have been a constant source of inspiration and strength.

I am also deeply grateful to my co-supervisor, Pr. Khalgui Mohammed for his significant contributions, particularly in shaping the article. His guidance has greatly enhanced the quality of my research and has been pivotal in this academic endeavor.

I am honored by the privilege of having my thesis evaluated by the esteemed members of the jury, Pr. Bennoui Hammadi, Pr. Chaoui Allaoua, and Dr. Samir Tigane. Their acceptance to examine this work is something I deeply appreciate, and I am grateful for their time and consideration.

Lastly, I wish to express my sincere thanks to all my teachers of the department.

Contents

List of Figures

List of Tables

General Introduction

Context and motivation:

In recent years, reconfigurable systems have gained prominence across various fields, including telecommunications, manufacturing [1], and embedded systems [2]. These systems dynamically adapt their structure and functionality in response to changing conditions or requirements. This adaptability allows for enhanced performance, resource efficiency, and responsiveness to user demands [3]. This trend reflects the broader shift toward intelligent and autonomous systems capable of reacting to a complex and rapidly evolving technological environment. With the increasing integration of cyber–physical architectures, reconfigurable systems now form the backbone of modern automation, robotics, and large-scale distributed applications [4]. However, as the complexity of these systems increases, so do the challenges associated with their design, implementation, verification and optimization. Understanding and addressing these challenges is essential for ensuring that reconfigurable systems operate reliably and efficiently in real-world scenarios. The growing importance of reconfigurable systems is tightly linked to the need for resilience and adaptivity in modern technological infrastructures. Systems are now expected not only to perform correctly under nominal conditions but also to maintain functionality in the presence of uncertainties, failures, and fluctuating workloads. As a result, reconfiguration has evolved from a desirable feature to a fundamental requirement in many applications, especially those demanding high levels of reliability and autonomy.

One of the primary challenges in managing reconfigurable systems is their inherent dynamic nature which introduces uncertainties and unpredictabilities that complicate both verification and optimization [5]. Ensuring that a system remains correct and efficient across all possible configurations requires rigorous verification techniques and effective optimization strategies. Furthermore, the interactions among various components in a reconfigurable system can lead to emergent behaviors that are difficult to predict, raising concerns about reliability and robustness. These challenges become even more pronounced when systems exhibit concurrent behaviors, distributed coordination, and time-dependent constraints. Even small structural changes can propagate and influence global system behavior, making informal reasoning insufficient and potentially dangerous in critical applications.

In addition to dynamic behaviors, other sources of complexity arise from architectural evolution, real-time constraints, and heterogeneous components operating concurrently. As systems evolve, new configurations may introduce interactions that were not originally anticipated, making it more difficult to guarantee correctness. Such challenges highlight the importance of adopting techniques capable of tracking structural changes, reasoning formally about their effects, and ensuring that essential properties are preserved throughout the reconfiguration process.

Additionally, the increasing scale of reconfigurable systems exacerbates these issues, leading to an exponential growth in the state space that must be analyzed. Traditional testing methods often fall short in thoroughly evaluating such vast and complex systems, resulting in undetected errors and vulnerabilities. Consequently, there is a pressing need for systematic approaches that can rigorously model and verify the behavior of these systems, ensuring their correctness and efficiency before deployment [6]. As systems grow in scale, the limitations of simulation-based or empirical validation become increasingly evident, highlighting the need for scalable mathematical tools supported by automation. However, while formal methods provide strong verification guarantees, they do not inherently address how to determine the best reconfigurations strategies, which is where optimization techniques become essential.

Formal methods [7] provide mathematically rigorous approach for specifying and verifying system behavior, ensuring compliance with safety, functional, and performance requirements.

Techniques such as model checking [8] and theorem proving [9] can uncover subtle errors and confirm that systems meet their specified requirements. These tools enable a systematic exploration of system behaviors, often revealing corner cases that manual analysis would fail to detect. They also foster a deeper understanding of system dynamics by allowing precise reasoning about constraints, dependencies, and execution scenarios. Despite their proven benefits, the application of formal methods to reconfigurable systems has been limited, primarily due to a lack of tailored methodologies that adequately address the unique characteristics of these systems.

Optimization plays a vital role in improving the adaptability and efficiency of reconfigurable systems. Techniques such as genetic algorithms [10], evolutionary strategies [11], and multi-objective optimization [12] have been widely used to determine optimal configurations for resource allocation, scheduling, and performance tuning. In reconfigurable manufacturing systems (RMSs), for example, selecting the best reconfiguration minimizes downtime and enhances productivity. Optimization is therefore indispensable when dealing with vast design spaces and conflicting objectives, especially in industrial settings where performance, cost, and reliability must be jointly considered. However, optimization alone does not provide formal guarantees about system correctness, making it necessary to integrate it with formal verification methods.

Despite the progress made in literature, many challenges remain unaddressed when dealing with reconfigurable systems. This leads to the questions that define the problematic of this research.

- How can we formally model and verify such systems especially the reconfiguration process itself?
- How can we verify correctness across a potentially large number of configurations while avoiding state-space explosion?
- What guarantees can be provided to ensure that essential properties such as safety, liveness, and performance are preserved after each reconfiguration?
- And finally, how can optimal configurations be identified efficiently while maintaining rigorous correctness guarantees?

These questions constitute the central problematic of this thesis and highlight the need for a unified methodology combining formal verification with effective optimization strategies.

This thesis aims to bridge the gap between formal methods and optimization by integrating formal modeling, verification, and optimization techniques into a unified framework for reconfigurable systems. Our objective is to develop robust methodologies that address reconfiguration while ensuring both correctness and efficiency. We seek also to establish a framework that not only enhances the reliability of reconfigurable systems, but also fosters their acceptance and implementation across various sectors.

To achieve these goals, we have identified several key objectives:

- to formulate a comprehensive modeling approach for reconfigurable systems that captures their dynamic behaviors and interactions,
- to develop verification techniques that ensure system correctness across various configurations,

- to enhance existing optimization solutions for these systems through formal methods, proposing new formalisms that improve performance and efficiency, and
- to provide insights into the integration of formal methods in the design process, promoting their adoption in industry practices.

Main contributions:

This thesis presents a comprehensive study of formal modeling and verification techniques tailored to the complexities of reconfigurable systems. The main contributions of this work are summarized as follows:

- ***Advanced Modeling Framework Using UPPAAL SMC for Mobile WSNs:*** Development of a robust modeling approach employing UPPAAL SMC and probabilistic automata to capture the dynamic behaviors and intricate interactions within mobile wireless sensor networks (MWSNs). This framework enables accurate representations of real-world scenarios, enhancing the reliability and performance of communication protocols.
- ***Comprehensive Verification Techniques with PCTL for IoT Protocols:*** Introduction of innovative verification methodologies that utilize probabilistic computation tree logic (PCTL) to ensure the correctness of IoT protocols, specifically focusing on the MQTT protocol. These techniques leverage formal methods to systematically identify and mitigate potential errors, significantly improving the trustworthiness of IoT applications.
- ***Novel Optimization Strategies with Genetic Algorithms for Reconfigurable Manufacturing Systems:*** Proposal of enhancements to the NSGA-II genetic algorithm by integrating formal methods techniques. This contribution allows for more effective optimization solutions in reconfigurable manufacturing systems, improving performance metrics and facilitating better decision-making in system design and operation.
- ***Integration of RONs for Reconfigurable Manufacturing Systems:*** Establishment of a formal modeling framework utilizing Reconfigurable Object Nets (RONs) to effectively model and verify reconfigurable manufacturing systems. This approach not only addresses the challenges of complexity and dynamic behavior but also provides a powerful tool for analysis and optimization.

Thesis structure:

The thesis is structured as follows:

- Chapter 1 provides a comprehensive overview of reconfigurable systems, detailing their definition, significance, and various applications. It explores the inherent challenges these systems face, such as complexity, reliability, and security concerns, while also reviewing current methodologies and frameworks used in the field.
- Chapter 2 is divided into two main parts. Part 1 covers the fundamental principles and importance of formal verification techniques, establishing the foundational knowledge required for understanding their application in reconfigurable systems. Part 2 delves into

genetic algorithms, their fundamentals, variants and multiobjective optimization solutions.

- Chapter 3 details in two parts the first contribution of the thesis, which involves using probabilistic timed automata for the formal modeling of mobile wireless sensor networks (MWSNs) and IoT protocols. It explains the application of UPPAAL SMC and PCTL for performance evaluation, presenting case studies and analyzing the obtained results.
- Chapter 4 focuses on the second major contribution, this chapter discusses the formal modeling, analysis and optimization of reconfigurable manufacturing systems using the high level Petri nets: Reconfigurable Object Nets (RONs). It also addresses optimization strategies enhanced by integrating RONs constructors in NSGA-II, detailing the methodology, results, and implications for practical applications.
- General conclusion concludes the thesis, summarises the contributions and discusses issues that are open for future research.

Chapter 1

Comprehensive Overview of Reconfigurable Systems

1.1 Introduction

In the rapidly evolving landscape of modern technology, the need for systems to adapt to dynamic environments has become more pronounced. From telecommunications and embedded systems to robotics and cloud computing, the demand for flexibility and adaptability in both hardware and software is crucial for achieving optimized performance, cost-efficiency, and scalability. Traditional static systems often fail to meet the demands of these fast-paced, variable environments, where constant reconfigurations are necessary to maintain efficiency, respond to failures, or upgrade functionality. As a result, the concept of flexibility in system design has become a pivotal factor in engineering solutions that can evolve in real-time.

This drive for adaptability has inspired a shift towards systems that can be reconfigured on demand, allowing them to evolve and self-optimize according to changing requirements. The ability to modify a system's structure, functionality, or behavior without extensive redesign or manual intervention is invaluable across numerous domains. In fields like aerospace, defense, automotive, and IoT, such capabilities lead to more resilient, efficient, and long-lived systems. Systems that exhibit reconfigurability can adjust their internal configurations, resource allocation, or operational modes based on environmental factors, system status, or external directives, offering a powerful tool to enhance performance and reduce operational costs.

The evolution of reconfigurable systems, however, brings with it several challenges, particularly in maintaining system reliability, performance, and security as configurations change. The increasing complexity of these systems requires novel approaches for managing their reconfigurability and ensuring that they operate as intended across various contexts and applications.

This chapter delves into the concept of reconfigurability, exploring why it is increasingly indispensable in modern systems. We examine the various ways in which reconfigurability manifests in different domains, highlighting its role in fostering innovation, efficiency, and adaptability. The remainder of this chapter starts with defining reconfigurable systems in detail in Section 1.2 and outline their key characteristics and applications, setting the stage for a deeper discussion on three reconfigurable systems which are: mobile sensor networks (MWSNs) in Section 1.7.1, Internet of Things (IoT) in Section 1.7.2, and reconfigurable manufacturing systems (RMSs) in Section 1.8.

1.2 Reconfigurable Systems (RSs)

Reconfigurable systems are systems designed to adapt dynamically by modifying their configuration, structure, or behavior in response to changing tasks, environments, or operational conditions. Unlike static systems, which are fixed in their design and function, reconfigurable systems incorporate mechanisms that allow for on-the-fly adjustments, enabling them to maintain or enhance their performance without requiring downtime or complete redesign [4][3].

Reconfigurability centers around the ability of a system to transition between multiple configurations during its operational lifecycle. According to [4], this concept is grounded in modularity and scalability, where individual system components or subsystems are designed to be easily interchangeable or adjustable. This modularity enables targeted changes to specific parts of the system without disrupting its overall functionality, a feature that is critical in domains like manufacturing, aerospace, and telecommunications.

The authors in [3] expand on this by emphasizing the system's ability to respond to real-time stimuli, whether these are environmental changes, workload variations, or system failures. For example, reconfigurable manufacturing systems dynamically alter their production setups to accommodate different product designs, while reconfigurable network routers adjust routing protocols to optimize data traffic.

1.2.1 Characteristics of Reconfigurable Systems

Several defining characteristics set reconfigurable systems apart from other adaptive or flexible systems [13][14] [4]:

- ***Modularity***: The design of reconfigurable systems emphasizes modular components, each capable of independent modification or replacement. This modularity supports scalability and ease of maintenance, as only specific modules need adjustment during reconfiguration.
- ***Adaptability***: Reconfigurable systems dynamically adapt their behavior to align with changing operational demands. For instance, an unmanned aerial vehicle (UAV) might alter its flight mode to optimize fuel efficiency under varying atmospheric conditions.
- ***Self-Optimization***: These systems are equipped with mechanisms to reallocate resources and optimize their performance under different workloads. A reconfigurable cloud computing platform, for instance, adjusts its computational resources to balance energy efficiency and processing speed.
- ***Fault Tolerance and Self-Healing***: Many reconfigurable systems integrate fault detection and recovery mechanisms, allowing them to reconfigure around failures. For example, in wireless sensor networks, reconfigurable nodes reroute data pathways to maintain network integrity despite individual node failures.
- ***Real-Time Operation***: Reconfigurable systems support dynamic reconfiguration during operation, enabling seamless transitions without downtime. This capability is particularly important in mission-critical applications such as avionics or industrial automation.

1.2.2 Reconfigurability and its Underlying Principles

The principles of reconfigurability provide a framework for understanding its role in system design [14]:

- Modularity and scalability: The ability to add, remove, or replace modules ensures that systems remain versatile and future-proof. Modular designs are particularly useful in industries like telecommunications, where network components must be frequently updated to meet new standards.
- Behavioral adaptation: Beyond physical changes, reconfigurable systems also adjust their behavior. For instance, a smart thermostat alters its temperature control algorithms based on user habits and external weather patterns, demonstrating logical reconfigurability.
- Incremental upgradability: Unlike static systems that require complete redesigns for upgrades, reconfigurable systems allow for incremental modifications, extending their lifecycle and reducing costs.
- Optimization of resources: By reallocating resources dynamically, reconfigurable systems ensure efficient operation, even under constrained conditions. For example, IoT devices reconfigure their communication protocols to conserve energy during low-battery scenarios.

1.2.3 Reconfigurability vs. Flexibility

While flexibility refers to a system's ability to a variety of tasks or conditions, reconfigurability relates to the mechanisms that allow a system to change its configuration or structure to meet these needs [15]. As described in [4], flexibility is the outcome or end goal, whereas reconfigurability provides the means or capability to achieve that goal. For example:

A flexible robotic arm might be programmed to perform multiple tasks like assembly or welding. A reconfigurable robotic arm, on the other hand, can alter its physical structure (e.g., by switching tools) or software to take on entirely new tasks beyond its initial programming.

This distinction underscores the importance of reconfigurability as the enabler that allows flexible systems to handle high variability and unpredictability in different environments. Table 1.1 illustrates various aspects of the comparison between reconfigurability and flexibility according to [4][15].

Table 1.1: Comparative table: Reconfigurability vs. Flexibility.

Aspect	**Reconfigurability**	**Flexibility**
Scope	Limited to specific configurations.	Broad adaptability across multiple dimensions.
Reversibility	Changes are reversible.	Changes may not be reversible.
Purpose and Goals	Focused on achieving predefined goals.	Focused on long-term adaptability.
Operational Scope	Limited set of achievable configurations.	Broad and unbounded range of adaptations.
Focus	Predefined configurations for specific needs.	Adaptability to a wide range of conditions.
Drivers	Adapting to multiple functions or conditions.	Addressing uncertainty and ensuring robustness.
Time Sensitivity	Requires timely reconfiguration.	Often for long-term adaptability.
Design Characteristics	Modular and adaptable elements.	Robustness, evolvability, change readiness.
Performance Impact	Ensures optimal performance in specific configurations.	Enhances capability to maintain or improve performance.
Examples	Modular robots, reconfigurable manufacturing.	Flexible manufacturing systems, adaptable product platforms.
Metrics and Evaluation	Evaluates reconfiguration effectiveness.	Measures redesign effort and adaptability.
Complexity	Involves reconfiguration mechanisms.	Complexity in adapting to unforeseen changes.
Change Mechanism	Requires physical or functional changes.	Involves abstract changes without physical reconfiguration.
Stability During Change	Controlled by robust systems.	Relies on inherent system robustness.
Research Challenges	Optimizing reconfiguration, minimizing costs.	Balancing cost, flexibility, and adaptability.

1.3 Classification of Reconfigurable Systems

The classification of reconfigurable systems can be based on several core principles, each defining the scope and functionality of the system:

1.3.1 Timing of Reconfiguration

Reconfigurable systems can be categorized by when reconfiguration occurs:

- Static reconfigurable systems: These systems can only be reconfigured before they begin operating. Once in use, they must continue in their initial configuration until halted. Example: programmable application-specific integrated circuits (ASICs) [4].
- Dynamic reconfigurable systems (DRS): These systems can reconfigure themselves while in operation, without downtime. DRSs are ideal for applications like UAVs, which may need to switch configurations during flight depending on mission requirements [3].

1.3.2 Control Mechanism

Another basis for classification is how reconfiguration is controlled:

- Manual reconfigurable systems: Human intervention is required for reconfiguration, typically seen in modular manufacturing systems where components or tools are physically replaced by operators [3].
- Automatic reconfigurable systems: These systems adjust configurations autonomously, often using control algorithms. A vehicle's control system that adapts driving modes based on sensor data is a prime example [4][3].

1.3.3 Levels of Abstraction

Reconfigurability can also be classified by the level at which reconfiguration occurs:

- Hardware reconfigurable systems: Systems like Field Programmable Gate Arrays (FPGAs) that allow for dynamic reconfiguration at the hardware level. These systems can alter their internal architecture on-the-fly, making them ideal for applications like real-time data processing [4].
- Software reconfigurable systems: Systems that adjust their operational logic or control algorithms. In IoT devices, software-driven reconfigurations allow for modifications to data transmission patterns or computational tasks depending on resource availability or external factors [3].

Figure 1.1 summarizes the classification of reconfigurable systems.

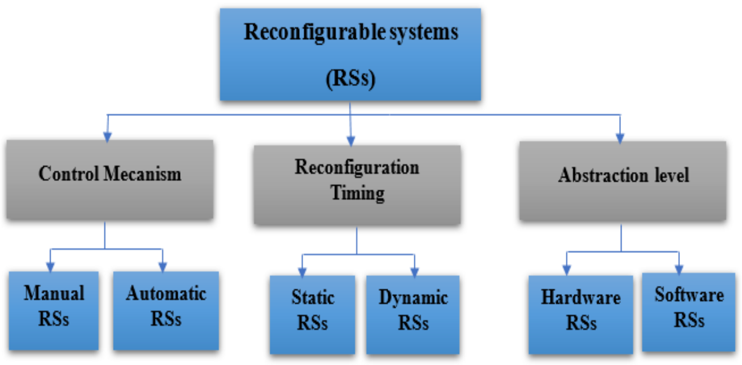


Figure 1.1: Classification of reconfigurable systems

1.4 Reconfigurable Systems across Various Domains

Reconfigurable systems are utilized in a wide range of domains, each exploiting the flexibility and adaptability of these systems to solve complex problems. Below is a briefly exploration of several key domains where reconfigurable systems are playing a critical role:

- **Aerospace and Defense**

  In the aerospace sector, reconfigurable systems enable unmanned aerial vehicles (UAVs) [16] to switch between flight modes (fixed-wing or rotor) to adapt to different environments. Reconfigurable radar systems [17] in defense also dynamically adjust frequencies to avoid jamming, ensuring secure communication.

- **Automotive Systems**

  Reconfigurable control systems [18] in the automotive industry are integral to the development of autonomous driving technologies. Advanced Driver Assistance Systems (ADAS) [19] reconfigure based on traffic conditions, switching between automated driving, lane-keeping, and parking assist. Hybrid vehicles also dynamically manage their power systems, switching between electric and combustion engines based on energy efficiency needs.

- **Manufacturing Systems**

  In manufacturing, Reconfigurable Manufacturing Systems (RMSs) [20] adapt production lines to handle different products or volumes. Reconfigurable systems allow for quick tool changes, modular robotic systems, and the integration of new tasks without requiring a full system overhaul.

- **Networking and Telecommunications**

  Reconfigurable systems in telecommunications use Software-Defined Networking (SDN) [21] and Network Function Virtualization (NFV) [22] to adaptively manage network traffic. These systems ensure bandwidth allocation, routing, and signal processing are optimized in real time, maintaining network reliability during heavy traffic loads or disruptions [3]. IoT devices and WSNs benefit greatly from reconfigurability, especially when managing energy consumption and data routing. Smart sensors in IoT reconfigure based on environmental conditions, optimizing their communication protocols and data transmission to conserve power. In WSNs, reconfigurable nodes adjust their behavior to account for node failure, optimizing network stability.

- **Healthcare and Medical Devices**

  Reconfigurable medical devices [23] such as adaptive wearables or smart prosthetics allow for real-time adjustments based on patient data. Devices like reconfigurable heart monitors change their operational modes based on detected irregularities, providing timely alerts to medical professionals.

- **Robotics**

  Reconfigurable robotic systems [24] are commonly used in industrial environments where tasks vary. Robots can switch between different tools or operational modes based on task requirements, such as welding, assembly, or painting. Modular robots can adapt their physical structure by adding or removing modules depending on the task's complexity.

As we explore these domains, two areas (reconfigurable manufacturing systems and networking) merit more in-depth discussion due to their significant impact on modern industry and technology. These fields are fundamental to the evolution of adaptable systems and present unique challenges and opportunities that will be examined in separate, dedicated sections.

1.5 Techniques and Methodologies for Reconfiguration

Reconfigurable systems rely on a variety of techniques and methodologies to achieve dynamic adaptation, ensuring efficient and effective transitions between configurations.

These techniques are integral to enabling reconfigurability across diverse domains, ranging from aerospace and telecommunications to manufacturing and healthcare. While each method has its strengths and applications, the choice of technique depends largely on the specific requirements and constraints of the system.

- ***Control Algorithms:***

  Control algorithms form the backbone of many reconfigurable systems, enabling real-time decision-making and execution of configuration changes. These algorithms typically monitor system performance and environmental conditions, identifying when and how reconfiguration should occur. For instance, in unmanned aerial vehicles (UAVs), control algorithms may trigger adjustments to flight configurations based on wind conditions or mission objectives [4]. Control algorithms are valued for their ability to execute decisions quickly and autonomously. However, their effectiveness often depends on the quality of the underlying logic and the availability of accurate sensor data. These factors can influence the robustness and adaptability of the system.

- ***AI-Driven Reconfigurations:***

  Artificial intelligence (AI) has increasingly been integrated into reconfigurable systems, particularly in applications requiring complex decision-making. Machine learning models, such as reinforcement learning algorithms, enable systems to predict optimal reconfigurations based on historical data and real-time inputs. For example, AI-driven techniques are employed in smart grids to reconfigure energy distribution dynamically, ensuring stability and efficiency under varying consumption patterns [3].

  AI techniques are particularly effective in environments with high variability and complexity. They allow systems to learn from past experiences and adapt their reconfiguration strategies over time. However, as with any heuristic-based approach, the outcomes are only as reliable as the training data and model assumptions.

- ***Model-Driven Development:***

  Model-driven development is another prominent approach used to design and implement reconfigurable systems. This methodology involves creating abstract models of the system to simulate and evaluate various configurations before they are deployed. These models are particularly useful in complex environments, such as avionics or industrial automation, where safety and reliability are paramount [4].

  Through simulations, engineers can anticipate the impact of reconfiguration decisions, ensuring that they meet system requirements and constraints. While model-driven development provides valuable insights, its effectiveness depends on the fidelity of the models and their ability to capture real-world dynamics accurately.

The variety of existing techniques highlights the versatility and adaptability of reconfigurable systems across diverse challenges and applications. Each methodology brings unique strengths, whether through rapid decision-making, data-driven optimization, or predictive modeling, allowing systems to balance performance, reliability, and efficiency in dynamic environments. However, as the complexity of reconfigurable systems increases, so does the need for more robust and comprehensive approaches.

Future advancements may benefit from integrating complementary methodologies that address current limitations and ensure system correctness under all conditions. In particular, the incorporation of formal modeling and verification techniques can play a pivotal role in bridging gaps, offering provable guarantees about system behavior, and providing a deeper understanding of configuration changes in mission-critical environments. Such techniques, when combined with existing methodologies, can enhance the reliability, safety, and scalability of reconfigurable systems, paving the way for more sophisticated and resilient solutions.

1.6 Future Challenges in Reconfigurable Systems

As reconfigurable systems become more widespread, several challenges need to be addressed:

- Complexity of design: Designing systems capable of dynamic reconfiguration, particularly in real-time, presents significant challenges in terms of software and hardware integration.
- Reliability: Ensuring that reconfigurable systems operate safely in critical industries (e.g., healthcare and aviation) requires rigorous testing and validation.
- Resource management: Managing the computational and energy resources needed for frequent reconfigurations is an ongoing challenge, especially in power-constrained environments like IoT and WSNs [4][3].

1.7 Reconfigurability in Networking

Reconfigurability in networking can be referred to the ability of a network to adapt dynamically to changes in its topology, structure, or operational requirements. This adaptability is essential for modern systems, especially in dynamic and resource-constrained environments [25]. In the context of Mobile Wireless Sensor Networks (MWSNs) [26] and Internet of Things(IoT) [27], mobility itself can be regarded as key form of reconfiguration [28][25].

MWSNs achieve reconfigurability by allowing mobile nodes to adjust their positions, enabling the network to self-organize and maintain functionality in response to environmental changes [25]. Similary, IoT systems leverage reconfigurable protocols to accommodate device mobility, scaling, and dynamic communication needs [28]. This section explores how reconfigurability manifests in these two domains, starting with MWSNs and their protocols, followed by IoT and its networking solutions.

1.7.1 Mobile Wireless Sensor Networks (MWSNs)

Mobile Wireless Sensor Networks (MWSNs) [26]extend traditional Wireless Sensor Networks (WSNs) by incorporating mobility into sensor nodes. This mobility enables them to adapt to dynamic environments and rapid topographic changes, enhancing their versatility. MWSNs consist of sensor nodes equipped with sensors (e.g., light, temperature, humidity), a radio transceiver, a microcontroller, and a power source [29]. In addition to these basic components, there are sensor nodes equipped with additional units, such as a localization system or a mobility unit, allowing them to move. Figure 1.2 illustrates a mobile wireless sensor network, showcasing the arrangement of mobile sensor nodes, their communication links, and the central base station for data collection. Applications of MWSNs span multiple domains, includ-

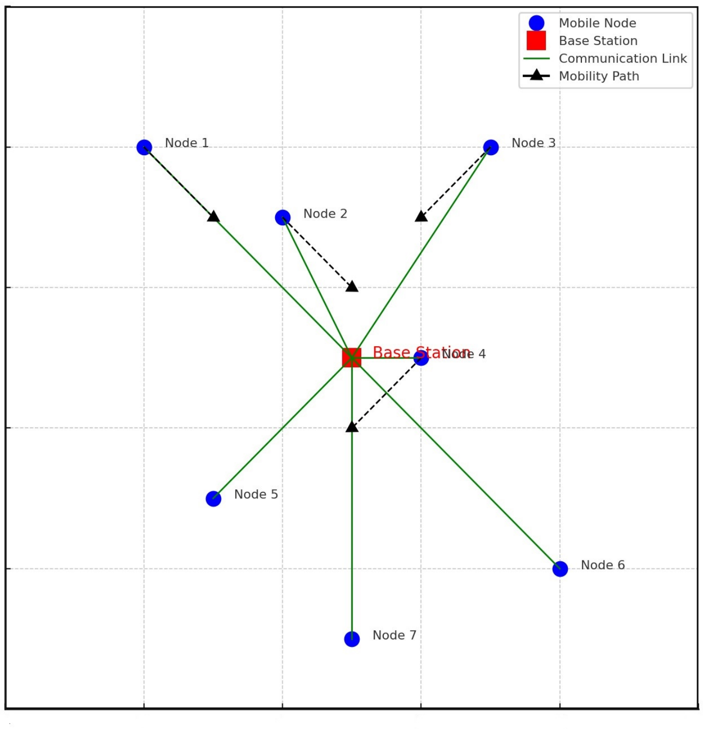


Figure 1.2: A mobile wireless sensor network

ing environmental monitoring, healthcare, military surveillance, and infrastructure protection. However, challenges such as limited energy, dynamic topology, and efficient data routing demand innovative solutions.

1. **Key features of MWSNs:**

   MWSNs offer unique features that distinguish them from traditional wireless sensor networks. These features enable MWSNs to adapt to dynamic environments, making them suitable for a wide range of applications. Some of the most notable features include [29]:

- **Mobility:** Enables nodes to reposition for optimal coverage and data collection.
- **Energy efficiency:** Protocols must address limited battery life and minimize energy use.
- **Dynamic topology:** Nodes frequently change positions, requiring adaptive network structures.
- **Scalability:** Supports large-scale deployments with numerous mobile nodes.
- **Versatility:** Applicable in various environments, such as undersea navigation, tactical surveillance, and environmental monitoring.

2. **Challenges in MWSNs:**

   Designing and deploying MWSNs involves overcoming a variety of challenges. These arise due to the inherent limitations of sensor nodes and the dynamic nature of their environments [29]. The key challenges are as follows:

   - ***Hardware limitations***

     Mobile sensor nodes in MWSNs face significant constraints in terms of processing power, memory, and energy resources. These nodes are typically battery-powered, making energy efficiency a critical design consideration. Simple microcontrollers and radios are used to reduce power consumption, but this often limits the complexity of operations that can be supported.

   - ***Dynamic topology***

     The mobility of nodes introduces frequent changes in network structure, making it difficult to maintain reliable communication paths. Network topologies must adapt in real-time to accommodate node movements, joinings, or departures. This dynamic nature places considerable strain on routing protocols, requiring them to be highly adaptable and efficient.

   - ***Environmental factors***

     Shared communication media in MWSNs increase the likelihood of interference, particularly in dense networks. This can degrade the quality of service (QoS) and reduce the reliability of data transmission. Additionally, topographic changes in deployment areas, such as hilly terrains or urban environments, further complicate network operations.

   - ***Routing protocol design***

     Frequent reconfiguration due to mobility significantly impacts routing protocols. Cluster-based routing, for instance, often requires nodes to repeatedly reconnect with cluster heads, increasing overhead and power consumption. At the same time, ensuring scalability, fault tolerance, and efficient data delivery remains a persistent challenge.

1.7.1.1 MWSNs' Protocols

In wireless sensor networks (WSNs), mobility management can be addressed at either the MAC (Medium Access Control) or network layer [30].

1. **MAC Layer Protocols**

Several approaches for mobile wireless networks have focused on adapting existing solutions designed for static scenarios. In many cases, mobile MAC protocols are derived from S-MAC, which serves as a foundation for handling mobility and energy efficiency [30]. The most notable protocols include:

- **S-MAC (Sensor MAC)** [31]: Optimizes energy usage by alternating between sleep and listening periods. Nodes synchronize their listening schedules, reducing idle time and energy consumption. S-MAC uses CSMA/CA for medium access and includes synchronization messages (SYNC) to maintain clock alignment.
- **MS-MAC (Mobile S-MAC)** [32]: An extension of S-MAC that supports mobile nodes by introducing shorter synchronized listening periods for neighboring nodes within a specific range. It aims to better track mobile nodes but suffers from similar limitations as S-MAC in dynamic environments.
- **MOB-MAC (Mobile MAC)** [33]: Designed to address frame loss and retransmission issues in MS-MAC, it adjusts frame sizes based on link quality. Smaller frames are used in poor conditions to reduce energy consumption and improve transmission reliability.
- **AM-MAC (Adaptive Mobility MAC)** [34]: Enhances S-MAC by introducing adaptive listening schedules and secondary listening periods. It organizes nodes into virtual clusters, allowing mobile nodes to quickly adapt to new schedules as they move between clusters. However, it incurs high energy usage for border nodes.
- **MD-SMAC (Mobility-aware Dynamic S-MAC)** [35]: Combines MS-MAC and DS-MAC to balance mobility support with energy efficiency, particularly for delay-sensitive applications. It adjusts the duty cycle based on the mobile node's energy level and optimizes the neighbor discovery process.
- **CFMA (Collision-Free Mobility Adaptive MAC)** [36]: Improves the back-off process by using predefined delays based on node priorities within a cluster. It adjusts delays according to signal strength from neighboring clusters, offering faster integration for new nodes.
- **MMAC (Mobility Adaptive Collision-Free MAC)** [37]: Adapts the frame time based on node mobility and network conditions, reducing synchronization delay. However, it assumes predictable node movement, which may not be valid in all real-world scenarios.

In addition to the synchronous solutions mentioned earlier, several asynchronous versions of the MAC protocol have been developed, including WiseMAC [38], B-MAC [39], X-MAC [40], RI-MAC [41], and A-MAC [42]. These asynchronous protocols offer a significant advantage by providing greater flexibility in handling node mobility compared to their synchronous counterparts. However, to effectively support node mobility, certain adaptations to these solutions are required. The MOX-MAC protocol [43] is one such adaptation. Furthermore, hybrid solutions combining both synchronous and asynchronous approaches have been introduced, such as the Mobile Adaptive MAC protocol (MAMAC) [44] and the Mobile Multimode Hybrid MAC protocol (MMHMAC) [45], which aim to leverage the strengths of both paradigms to enhance performance in dynamic environments.

2. **Network Layer Protocols**

In MWSNs, the network layer protocols (routing protocols) are essential for adapting to dynamic topologies caused by node mobility. These protocols ensure efficient routing, maintain connectivity, and enhance the network's reconfigurability under resource constraints. According to [29] some of these protocols are:

- LEACH-M (Low-Energy Adaptive Clustering Hierarchy-Mobile):
  An adaptation of the LEACH protocol [46] for mobile nodes, LEACH-M [47] addresses re-clustering issues caused by node movement. While energy-efficient clustering is beneficial, frequent switching between clusters can increase overhead.
- Angle-Based Dynamic Source Routing (ADSR):
  ADSR [48] extends the Dynamic Source Routing (DSR) protocol [49] by incorporating location information. This ensures packets are routed optimally toward the sink.
- Zone-Based Routing Protocol (ZRP):
  ZRP [50] uses geographical data to define zones and update routing tables dynamically. This reduces communication overhead and enhances scalability in mobile environments.
- Geographical Opportunistic Routing (GOR):
  GOR [51] divides the network into logical segments and forwards data opportunistically, optimizing routing in mobile scenarios.
- Multipath Data Center-Based Routing (MDCR):
  MDCR [52] provides fault tolerance by establishing multiple paths for data transmission, minimizing delays and congestion.

1.7.2 Internet of Things and Reconfigurability

The Internet of Things (IoT) [27] is a network of interconnected devices capable of sharing data and performing tasks autonomously. The ability of these devices to adapt to changing environments, scale efficiently, and ensure seamless communication is crucial. Reconfigurability plays a key role in enabling IoT systems to dynamically adjust to new conditions, such as adding or removing devices, handling mobility, and optimizing resource usage. This section explores the protocols that support reconfigurability in IoT networks, focusing on how these protocols allow IoT systems to adjust to varying requirements and conditions.

1.7.2.1 Definition of IoT

The Internet of Things (IoT) [27] refers to a vast network of interconnected devices that communicate and exchange data over the internet. These devices range from simple sensors and actuators to advanced computing systems, all of which are equipped with embedded technologies that enable interaction with the physical and digital environments. IoT has become an essential enabler of smart systems, where devices collect, process, and transmit data autonomously, fostering automation and real-time decision-making in various domains like healthcare, transportation, agriculture, and industry [53].

According to [53], IoT has revolutionized the way physical entities interact by integrating advanced connectivity protocols, low-power communication technologies, and data management tools to create efficient, scalable, and intelligent ecosystems.

At its core, IoT transforms traditional objects into "smart" devices, enabling autonomous operations without human intervention. For instance, IoT-connected buildings, cities, and workplaces utilize protocols like Zigbee, LoRaWAN, and BLE to achieve seamless communication and reconfigurability in response to environmental changes [54].

1.7.2.2 Characteristics of IoT

IoT is defined by several distinct characteristics that make it a transformative technology [55]:

- Ubiquitous connectivity: IoT enables global connectivity through seamless device integration.
- Dynamic adaptability: IoT systems can reconfigure themselves to accommodate device mobility, scaling, or failures.
- Energy efficiency: Lightweight communication protocols and optimized hardware design ensure extended device operation.
- Interoperability: IoT systems bridge diverse devices, supporting different standards and communication technologies.
- Real-Time analytics: Data collected by IoT devices is analyzed in real time, enabling quick responses to changing conditions.

1.7.2.3 IoT Protocols Supporting Reconfigurability

Reconfigurability in IoT relies on protocols designed to adapt dynamically to evolving network conditions, ensuring seamless communication and optimal performance. Notable among these protocols are [53]:

1. MQTT (Message Queuing Telemetry Transport):

   MQTT [56] is a lightweight publish/subscribe protocol ideal for constrained devices. Its flexibility allows dynamic topic management and adjustable QoS levels, ensuring reconfigurability in data delivery and device integration.

2. CoAP (Constrained Application Protocol):

   CoAP [57] is a restful protocol for resource-constrained IoT systems. It supports dynamic resource discovery, enabling devices to integrate seamlessly into existing networks while conserving energy.

3. LoRaWAN (Long-Range Wide Area Network):

   The adaptive data rate mechanism of LoRaWAN [58] dynamically adjusts transmission power and data rates, optimizing energy use and supporting network scalability.

4. Zigbee:

   A widely adopted protocol for low-power IoT applications, Zigbee [59] [60] excels in mesh networking. Its self-healing capabilities allow the network to adapt to device failures or topology changes.

5. BLE (Bluetooth Low Energy)

   BLE [61] enables short-range communication with adjustable connection parameters, ensuring real-time reconfiguration for energy efficiency and low-latency communication.

1.7.2.4 IoT Applications

IoT has expanded into numerous domains, driving innovation and efficiency across industries [53]:

- Smart cities: IoT technologies enhance urban living by optimizing traffic management, energy usage, and public safety.
- Healthcare: IoT devices enable remote patient monitoring, real-time health tracking, and predictive diagnostics.
- Industrial IoT (IIoT): Industrial applications benefit from IoT-driven automation, predictive maintenance, and supply chain optimization.
- Agriculture: IoT sensors monitor soil conditions, weather patterns, and crop health, improving agricultural efficiency.

1.8 Reconfigurable Manufacturing Systems (RMSs)

The continuous pressure for faster production, lower costs, and higher customization in manufacturing has given rise to the need for systems that can adapt quickly to new demands. Reconfigurable Manufacturing Systems (RMSs) [20] have emerged as the answer to this challenge. RMS are designed to provide not only flexibility but also scalability and customization to suit varying market conditions. In this chapter, we will explore the foundations of RMS, their components, applications across different industries, and their critical role in the future of manufacturing.

1.8.1 Definition of Reconfigurable Manufacturing Systems (RMSs)

Reconfigurable Manufacturing Systems (RMSs) [20] are advanced manufacturing setups purposefully designed to adapt swiftly to changes in production requirements. Unlike traditional systems, which often struggle to balance flexibility and efficiency, RMS provides the capability to adjust production capacity and functionality in response to fluctuating market demands, new product designs, or technological advancements. The concept of RMS revolves around creating systems that are inherently agile, enabling manufacturers to optimize production processes with minimal downtime or costs ([20], [62]).

The emergence of RMS in the mid-1990s was driven by the limitations of existing paradigms such as Dedicated Manufacturing Lines (DML) and Flexible Manufacturing Systems (FMS). DML, while efficient for mass production, lacked the flexibility needed for product variation or fluctuating demand. Conversely, FMS offered flexibility but at the expense of higher costs and reduced throughput. RMS bridges this gap, combining the high efficiency of DML with the adaptability of FMS, making it ideal for industries facing frequent product changes and unpredictable market dynamics ([20], [62]).

Conceptually, RMS is built to enhance responsiveness and sustainability in manufacturing. It allows manufacturers to adapt their production processes to varying requirements with minimal disruption. By focusing on rapid reconfigurability, these systems support competitive pricing, optimize resource use, and reduce the costs associated with traditional system overhauls. As highlighted by [20], RMS aims to create a "live factory," where manufacturing setups are not static but evolve dynamically to meet the ever-changing demands of global markets ([62]).

1.8.2 RMS Key Characteristics

Reconfigurable Manufacturing Systems (RMS) are characterized by distinct features that enable their adaptability and responsiveness to market demands. These key characteristics set RMS apart from traditional manufacturing systems and are integral to their functionality and design. Each characteristic plays a critical role in ensuring the system's flexibility and scalability while maintaining efficiency and cost-effectiveness.

- **Scalability** Scalability refers to the ability of an RMS to adjust its production capacity by adding or removing resources as needed. This feature allows manufacturers to respond effectively to increases or decreases in demand without requiring a complete overhaul of the manufacturing system. For example, in automotive production, additional machines can be integrated into an RMS to handle a surge in vehicle orders, ensuring rapid and cost-effective capacity adjustments ([62]).

- **Modularity** Modularity involves designing the system as a collection of independent and interchangeable units or modules. Each module performs a specific function and can be rearranged, replaced, or upgraded without disrupting the entire system. Modularity not only simplifies maintenance and upgrades but also reduces reconfiguration time, making it a cornerstone of RMS design ([63], [62]).

- **Convertibility** Convertibility is the ability of an RMS to change its functionality to produce new products or variations within a product family. This characteristic ensures that the system remains relevant in the face of evolving product designs and customer preferences. Convertibility is achieved through flexible hardware and software configurations that can be reprogrammed or adapted as needed ([20], [62]).

- **Diagnosability** Diagnosability enables real-time monitoring and rapid identification of issues within the system. Integrated diagnostic tools allow RMS to maintain high product quality by detecting and addressing faults or deviations promptly. This characteristic minimizes downtime and ensures consistent production standards, a critical requirement in industries like aerospace and electronics ([62]).

- **Customization** Customization in RMS refers to the system's ability to focus on producing specific product families with high efficiency. Unlike Flexible Manufacturing Systems (FMS), which handle a wide range of products, RMS optimizes its flexibility within a defined product scope, balancing adaptability and production efficiency ([20]).

- **Integrability** Integrability is the ease with which new modules, machines, or technologies can be integrated into the existing system. This characteristic ensures seamless upgrades and expansions, allowing manufacturers to keep pace with technological advancements and changing market requirements ([64], [62]).

These six core characteristics work together to provide RMS with the adaptability and responsiveness necessary to thrive in dynamic manufacturing environments. They enable cost-effective reconfiguration, maintain high-quality production, and ensure that systems can evolve to meet future demands. RMS's unique combination of these features makes it a powerful tool for industries requiring both flexibility and efficiency in their production processes.

1.8.3 Types of Reconfigurable Machines in RMS

Reconfigurable machines are essential components of Reconfigurable Manufacturing Systems (RMS), providing the flexibility to adjust to various production needs. These machines are designed to support different manufacturing tasks and can be rapidly reconfigured to accommodate changes in product design, production volume, or process type. Below are the key types of reconfigurable machines used in RMS, with detailed descriptions of each type [64].

1. **Reconfigurable Machine Tools (RMTs)**

   Reconfigurable Machine Tools (RMTs) are versatile pieces of equipment used in machining operations such as turning, milling, and drilling. These machines are designed with modular components, such as adjustable tooling, fixtures, and spindle heads, that allow them to be rapidly reconfigured for different parts within a product family. RMTs enable manufacturers to perform multiple operations on various products without requiring extensive setup times or new equipment for each product change.

   RMTs are especially useful in industries like automotive and aerospace, where product designs change frequently and high levels of precision are required. By leveraging the flexibility of RMTs, manufacturers can reduce downtime and setup costs, making these tools an essential part of RMS.

2. **Reconfigurable Inspection Machines (RIMs)**

   Reconfigurable Inspection Machines (RIMs) are designed for quality control within the RMS. These machines are equipped with advanced sensors, such as laser scanners, cameras, and vision systems, to perform in-line inspections of parts and products as they move through the production process. What distinguishes RIMs is their modular nature, allowing them to be quickly reconfigured to inspect different parts within a product family, thereby ensuring product quality without the need for separate dedicated inspection machines.

   It is noted that RIMs can be adapted to accommodate different product geometries or inspection requirements, making them highly adaptable to fluctuating production needs. This flexibility helps reduce the need for additional inspection lines when product variants change, improving overall system efficiency and reducing inspection costs.

3. **Reconfigurable Assembly Machines (RAMs)**

   Reconfigurable Assembly Machines (RAMs) are used to assemble different products or product variants in a flexible and efficient manner. These machines typically use modular workstations, robotic arms, and interchangeable tools that can be reconfigured to handle various assembly tasks without requiring dedicated systems for each product type. RAMs are essential in industries with high product variation, such as consumer electronics, where frequent model changes or customizations are common.

   For example, in the assembly of electronic devices, RAMs can be adapted to assemble smartphones, tablets, or other devices with minimal downtime between product changes.

This allows manufacturers to maintain high throughput and flexibility, ensuring that production lines can meet changing market demands.

4. **Reconfigurable Fixtures (RMS-F)**

   Reconfigurable Fixtures are critical components in RMS that hold parts in place during machining, assembly, or inspection. These fixtures are modular and can be adjusted to accommodate parts of different shapes and sizes, making them highly adaptable for producing parts within a product family. The use of reconfigurable fixtures helps manufacturers save costs by allowing the same fixtures to be used for multiple products, eliminating the need for custom fixtures for each new product design.

   Reconfigurable fixtures are particularly important in precision manufacturing industries such as aerospace, where parts often have complex geometries and require high accuracy during production. These fixtures can be easily modified to secure various parts in place, ensuring that each part is accurately positioned for processing.

1.8.4 Performance Objectives in RMS Optimization

The optimization of Reconfigurable Manufacturing Systems (RMS) is evaluated using various objective functions that are categorized into two distinct levels: machine-level and system-level [64]. These levels address different aspects of RMS performance:

1. Machine-level objectives

   At the machine level, objective functions are designed to enhance the performance of individual reconfigurable machines, such as Reconfigurable Machine Tools (RMTs). Key objectives include:

   - Cost minimization: This includes reducing costs related to: (i) The usage and maintenance of machines, tools, and modules. (ii) Energy consumption incurred during machining tasks. (iii) Reconfiguration costs, which cover changes in machine setups, such as reorienting tools and replacing modules.
   - Time optimization: Time-related objectives focus on: (i) Minimizing the completion time for producing a unit product. (ii) Reducing the makespan, which represents the total time needed to process all products in a production cycle. (iii) Lowering the reconfiguration time, which includes the time required to adjust machine setups and change tools.
   - Enhancing machine reconfigurability: This objective evaluates the machine's ability to switch configurations efficiently. It measures: (i) The number of feasible alternative configurations. (ii) The effort needed to achieve these configurations, often quantified by the number of modules that need to be added, removed, or repositioned.
   - Improving operational capability: This ensures that machines can handle a diverse set of manufacturing tasks without requiring extensive modifications.

2. System-level objectives

   At the system level, optimization focuses on the overall performance of the RMS, encompassing all its components such as machines, workstations, and material handling systems. Key objectives include:

- Cost minimization: System-level cost objectives are categorized as: (i) Capital Costs: Expenses for purchasing machines, modules, and tools. (ii) Operating Costs: Costs incurred during production, such as energy and maintenance. (iii) Reconfiguration Costs: Costs related to adding, removing, or repositioning machines and equipment.
- Time optimization: Reducing delays and optimizing the overall production schedule are critical. This includes minimizing: (i) Total production time for different batches. (ii) Delays caused by reconfiguration and material handling.
- Maximizing energy efficiency: This involves reducing energy consumption across the RMS by optimizing energy use during machining, transportation, and reconfiguration processes.
- Enhancing system flexibility and scalability: (i) Flexibility: The ability of the RMS to adapt to new product designs or variations with minimal disruptions. (ii) Scalability: Adjusting the production capacity of the RMS to align with fluctuating market demands.
- Improving system utilization: Ensuring that all resources, including machines and workstations, are used efficiently.

The objectives at both levels are often interdependent. For example, reducing reconfiguration time at the machine level contributes to higher throughput at the system level. Similarly, minimizing costs at the system level requires efficient utilization of individual machines. This interconnected nature underscores the importance of defining clear and measurable performance objectives for RMS.

1.9 Conclusion

In this chapter, we provided a comprehensive overview of reconfigurable systems, exploring their definition, characteristics, and the principles underlying their reconfigurability. We compared reconfigurability with flexibility and discussed the various classifications of reconfigurable systems across different domains. Emphasis was placed on the wide-reaching applications of reconfigurability, particularly in networking domains such as Mobile Wireless Sensor Networks (MWSNs) and the Internet of Things (IoT), as well as in the context of Reconfigurable Manufacturing Systems (RMS).

As we transition to the next chapter, the focus will shift toward formal methods and optimization techniques. This will include a detailed exploration of the formal modeling, verification, and optimization approaches essential for ensuring the reliability and performance of reconfigurable systems. Through the application of these methods, we can address the complexities of reconfigurability and move towards more robust and efficient system designs.

Chapter 2

Formal Methods and Genetic Algorithms

2.1 Introduction

In the field of system design, reconfigurability has become a core requirement, enabling systems to adapt dynamically to new conditions, changes in requirements, or different environmental constraints. This adaptability is crucial for industries such as manufacturing, networking, and embedded systems, where systems must cope with uncertainties and variability in their operating environments.

To manage this complexity, two powerful approaches come into play: formal methods [7] and genetic algorithms (GAs). Formal methods provide a rigorous mathematical framework to model and verify systems, ensuring correctness and predictability. These methods are discussed in Section 2.2, which gives an overview of the techniques and details their application in reconfigurable systems. Subsection 2.2.2 focuses on formal verification using model checking, explaining how this approach ensures system correctness through exhaustive state-space exploration. Subsection 2.2.3 addresses properties specification, outlining how to formally specify desired system behaviors and constraints. In Subsection 2.2.4, formal modeling methods are examined, demonstrating how abstract models can predict and verify reconfiguration behavior. On the other hand, GAs offer a robust optimization framework inspired by natural selection to find near-optimal solutions in large search spaces. Section 2.3 introduces genetic algorithms, starting with their fundamentals in Section 2.3.1. Section 2.3.3 discusses variants of GAs, exploring different adaptations and improvements for specific applications. Finally, Subsection 2.3.4.1 focuses on multiobjective genetic algorithms, detailing how they handle optimization tasks with multiple competing objectives. This chapter explores these two approaches in depth and demonstrates how they can be applied to reconfigurable systems.

2.2 Formal Methods

Formal methods [7] are mathematical techniques used for the specification, development, and verification of systems, particularly in software and hardware design. Unlike empirical testing, formal methods provide a rigorous framework that guarantees certain properties of the system. In reconfigurable systems, which need to adapt dynamically, formal methods ensure that these adaptations do not compromise the correctness or performance of the system. The key characteristics of formal methods are:

- **Precision:** Formal models eliminate ambiguity in system specifications.
- **Verification:** Using formal verification, one can mathematically prove properties like safety, liveness, and performance bounds.
- **Scalability:** While formal methods can be computationally expensive, they are indispensable for verifying large systems with high complexity.

Formal methods have been employed extensively in fields like cyber-physical systems, embedded systems, and reconfigurable manufacturing to ensure reliability in systems that must frequently adapt to new configurations. This section will explore formal methods broadly, then delve into specific techniques like model checking [8], automata [65], and Petri nets [66], with a special focus on Reconfigurable Object Nets (RONs) [67]. We will also cover the key aspects of formal verification, formal modeling, and properties specification formalisms.

2.2.1 Formal Methods Overview

Formal methods are employed to ensure that systems behave as intended and meet their requirements. They provide the means to define system behaviors unambiguously, verify these behaviors rigorously, and refine system designs in an incremental and controlled manner [68]. Formal methods can be divided into [69]:

1. **Specification:**

   The specification is the act of defining a system's behavior using formal languages. By using mathematical notations, formal specification eliminates ambiguity, providing a clear understanding of how the system should operate under various conditions. Several specification formalisms are widely used [70], [71]: (i) *Z Notation:* [72] Based on set theory and predicate logic, Z is often used for data-intensive systems. It defines the system's states and operations in a formalized manner, ensuring consistency between what is specified and what is ultimately implemented. (ii) *B Method:* [73] This method formalizes systems as abstract machines. Operations are modeled mathematically to transform system states, and the B method emphasizes refinement, incrementally transforming abstract models into implementable systems. (iii) *VDM (Vienna Development Method):* Like Z, VDM focuses on specifying software and hardware systems through abstraction and modularity. VDM is used for complex systems, especially in industrial settings [74]. (iv) *Temporal Logic:* Temporal logic, particularly Linear Temporal Logic (LTL) [75] and Computational Tree Logic (CTL) [76], is used to specify the ordering of events and behaviors in reactive systems, such as real-time or distributed systems.

2. **Verification:**

   The verification is the process of ensuring that a system adheres to its specification. Unlike traditional testing, which can only cover a limited number of scenarios, formal verification guarantees that all possible states of a system have been accounted for. Formal verification methods include:

   - Model Checking: An automated technique where a model of the system is exhaustively explored to check for properties, typically expressed in temporal logic. Model checking ensures that every possible execution of a system meets the specified requirements. The key Tools are: UPPAAL [77], PRISM [78].
   - Theorem Proving [9]: is a formal verification technique used to ensure that a system or program adheres to its specification through rigorous logical deduction. It involves defining the system's behavior in formal logic, generating proof obligations that must be satisfied, and using theorem provers to attempt to prove these obligations. The process may be automated or require human intervention, depending on the complexity of the proof [79]. There are different types of theorem provers: automated theorem provers attempt to automatically prove properties, interactive theorem provers (e.g., Coq [80], Isabelle [81]) require human guidance for complex proofs, and decision procedures specialize in specific kinds of logical formulas, such as arithmetic or bit-vectors.

     The advantages of theorem proving include providing strong mathematical guarantees of correctness, handling infinite state spaces (useful for verifying systems with recursive or unbounded behaviors), and being applicable across various domains such as software, hardware, and cryptography. However, it also has disadvantages, such as high computational complexity, scalability issues for large systems, the need for expert human intervention in interactive tools, and steep learning curves for users unfamiliar with the tools.

3. **Refinement:**

   Refinement transforms an abstract system specification into a detailed design or implementation. The key goal is to ensure that each step of refinement preserves the properties established in the original specification. There are two types of refinement: *(i) Data Refinement:* transforms abstract data structures into concrete implementations. For example, an abstract set may be implemented as a list or array in the final system. *(ii) Operation Refinement:* Replaces high-level operations with more concrete steps. An abstract operation like "assemble" in a manufacturing system could be refined into specific robot arm movements and conveyor belt actions.

The three aspects of formal methods (specification, verification, and refinement) are closely linked. Specification provides the foundation by defining what the system should do. Verification ensures that the system actually behaves as specified. Refinement allows the abstract model to be gradually transformed into a concrete system while preserving correctness.

In reconfigurable systems, this link is crucial. A formal specification might define how the system should behave across different configurations. Verification techniques such as model checking or theorem proving ensure that these behaviors are maintained, even as the system changes. Finally, refinement ensures that each component is implemented correctly, allowing the system to evolve without introducing errors.

2.2.2 Formal Verification Using Model Checking

Model checking [82][8] is one of the most widely used formal verification techniques, enabling the automatic verification of a system's operation by systematically exploring all possible states to ensure that specific properties hold. This technique is particularly beneficial for verifying concurrent, distributed, or reactive systems, where numerous states and transitions exist. The model checking process involves several key steps:

- **Modeling the system:** The system is represented as a finite-state machine or automaton, capturing the states and transitions between them.
- **Defining properties:** The properties to be verified are expressed in temporal logics, such as Linear Temporal Logic (LTL) or Computation Tree Logic (CTL). These properties specify how the system should behave over time, covering safety, liveness, and fairness requirements.
- **Verification process:** A model checker explores the entire state space of the system to verify whether the defined properties hold. If a property is violated, the tool provides a counterexample showing how the error occurs.
- **Error correction:** Counterexamples provided by model checking allow developers to trace and fix bugs in the system's design.

The model checking approach has been extensively applied across fields such as communication protocol verification, multimedia applications, and computer security, highlighting its

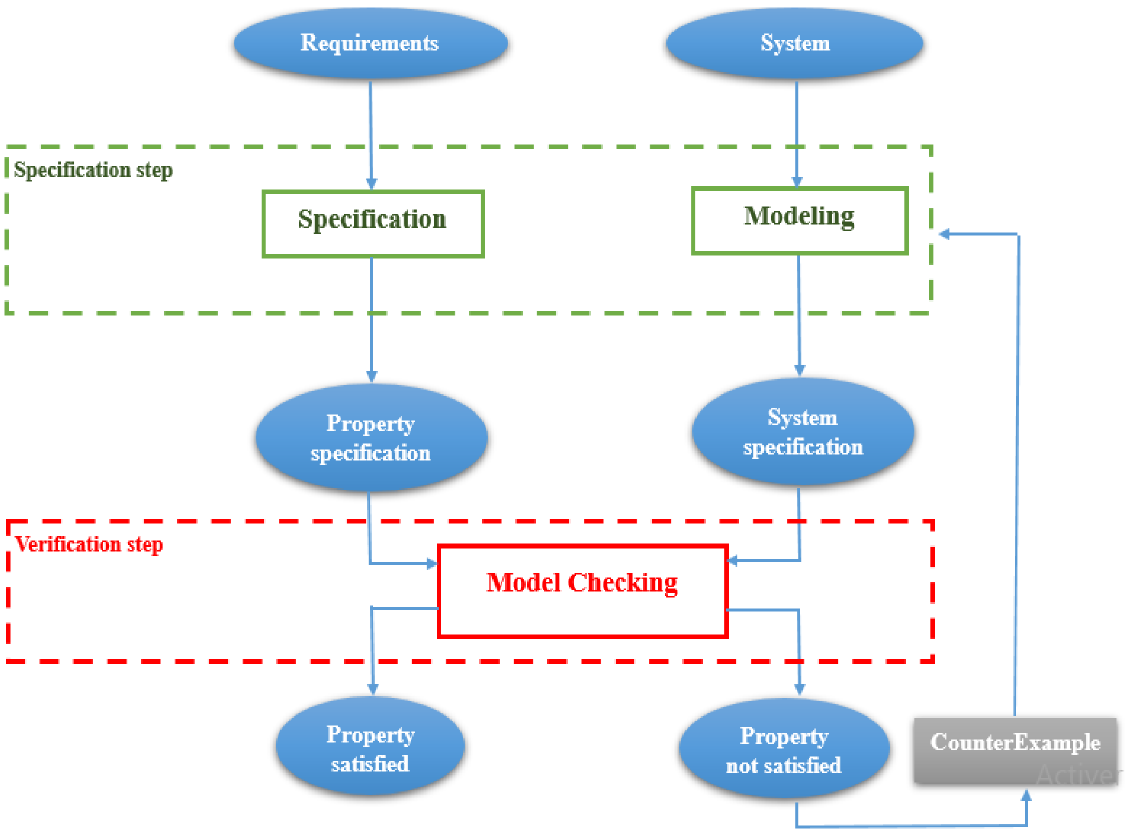


Figure 2.1: Model checking procedure.

versatility and effectiveness. The overall verification process consists of aforementioned steps, as illustrated in Figure 2.1.

The advantages of model checking are:

- **Exhaustive:** It explores all possible system states.
- **Automated:** Once the model and properties are defined, the process is fully automated.
- **Debugging:** Counterexamples help identify and resolve issues efficiently.

However, the disadvantage of model checking is the problem of combinatorial state space explosion, caused by the exponential growth of the state space size of a concurrent system with respect to the number of processes and the number of components per process [83]. For example, considering a system composed of n processes, each having m states, the asynchronous composition of these processes will have $m \times n$ states [83].

To mitigate this combinatorial state explosion, several approaches have been proposed [84]:

1. Symbolic Model Checking [85]: Symbolic model checking is a more efficient technique than classical model checking. It considers a large number of states simultaneously in a single step, rather than traversing accessible states one at a time.

2. Bounded Model Checking [86]: A Boolean formula is constructed that is satisfiable if and only if there exists a counterexample of length k. By incrementing the bound k, additional counterexamples can be searched. After a certain number of iterations, we can conclude that no counterexample exists.

3. Probabilistic Model Checking [87]: Various approaches have been proposed for building tools for probabilistic model checking. PRISM [78] is an example of a probabilistic model checking tool that can be used for accessibility analysis and protocol verification. Probabilistic model checking mitigates the state-explosion problem by using symbolic data structures, such as Multi-Terminal Binary Decision Diagrams (MTBDDs), to represent Markov chains and Markov decision processes compactly. Instead of enumerating every individual state and transition, probabilistic model checking groups similar probabilistic behaviors together and represents transition matrices symbolically. This allows the model checker to analyze very large systems by solving sets of linear equations over these compact symbolic structures rather than exploring the full underlying state graph.

4. Statistical Model Checking: Statistical model checking (SMC) [88][89] is an extension of classical model checking based on stochastic simulation. This extension was proposed to express the probabilistic aspects necessary for evaluating the performance of certain systems. SMC is an approach for the automatic verification of quantitative properties with a probabilistic aspect. An SMC model checker generates a number of random simulations (executions) and then uses algorithms to make probabilistic decisions about the validity of certain properties.

   The advantage of the statistical approach lies in its complexity growing proportionally with the size of the model. Indeed, it does not store the entire state graph when generating a path (sample); it only stores the current state. In SMC, there must always be an algorithm for generating random simulations (an algorithm used in the Uppaal tool is presented in the paper [90]).

   This approach is implemented in numerous tools such as COSMOS [91], UPPAAL [77], and VESTA [92].

2.2.3 Properties Specification Formalisms

Properties specification formalisms are used to express the requirements a system must meet during verification. In model checking, the most commonly used formalisms are LTL (Linear Temporal Logic) and CTL (Computation Tree Logic). These two logics have extensions designed to explicitly incorporate time and probabilities, such as PLTL (Probabilistic LTL), TLTL (Timed LTL), PCTL (Probabilistic CTL), TCTL (Timed CTL), and PTCTL (Probabilistic Timed CTL) and Other important formalisms include: (i) Metric Temporal Logic (MTL): An extension of LTL that allows real-time constraints, making it suitable for real-time systems, (ii) Alternating-time Temporal Logic (ATL): An extension of CTL, used for multi-agent systems where the actions of different agents influence system states.

1. **Linear Temporal Logic (LTL):**

   LTL [75] is used to specify properties over linear sequences of events, making it useful for verifying sequential system behaviors. Examples of LTL properties include: (i) Safety: "The system never reaches a deadlock state.". (ii) Liveness: "Every request will eventually be granted.

   An LTL formula ϕ has the following syntax:

   $$\phi ::= p|(\neg\phi)|(\phi \wedge \phi)|(\phi U \phi)|(G\ \phi)|(F\ \phi)|(X\ \phi)$$

   where:

- p is an atomic proposition,
- X_ϕ: ϕ holds next time,
- F_ϕ: ϕ holds sometime in the future,
- G_ϕ: ϕ holds globally,
- ${}_pU_q$: p holds until q holds.

2. **Computation Tree Logic (CTL):**

   Computational Tree logic [76] allows for the expression of properties related to execution trees that represent the evolution of a system. CTL formulas are expressed using the following language:

   - ***Atomic propositions:*** $p, q, \ldots, \text{true}, \text{false}$ which are directly interpreted at a state.
   - ***Boolean connectors:*** $\neg$ (negation), $\wedge$ (logical and), $\vee$ (logical or), $\Leftrightarrow$ (equivalence), $\Rightarrow$ (implication).
   - ***Temporal connectors:*** $EFp|EGp|EpUq|EXp|AFp|AGp|ApUq|AXp$ where E and A are path quantifiers.

   (a) ***Syntax of a CTL Formula:***

   Let φ and Ψ denote two CTL formulas. CTL formulas can be interpreted as follows:
   $\varphi, \Psi ::=$ I $\neg\varphi|\varphi \wedge \Psi|\varphi \vee \Psi|\varphi \Leftrightarrow \Psi|\varphi \Rightarrow \Psi$: propositional logic operators,

   $|EF\varphi$: There exists a sequence of states (operator E) that leads to a state where property φ holds (Finally operator F),

   $|EG\varphi$ There exists a sequence of states where property φ is always satisfied (Globally operator G),

   $|E\varphi U\Psi$: There exists a sequence of states where property φ is satisfied until property Ψ holds,

   $|EX\varphi$: There exists a sequence of states where there is a state whose successor state (Next operator X) satisfies property φ,

   $|AF\varphi$: For every sequence of states (operator A), there exists a state where property φ holds,

   $|AG\varphi$: For every sequence of states, property φ is always satisfied,

   $|A\varphi U\Psi$: For every sequence of states, property φ is satisfied until property Ψ holds,

   $|AX\varphi$: Property φ is satisfied in all immediately succeeding states of the current state.

   (b) ***CTL Semantics:***

   Formally, CTL is interpreted over the states of a transition system (which characterizes the trees stemming from these states).

   - ***Labeled Transition System*** S***:*** A transition system S is a tuple $(Q, q_0, A, \rightarrow, L)$ such that: Q is a (finite) set of states, $q_0 \in Q$ is the initial state, A is a (finite) alphabet of actions, $\rightarrow \subseteq Q \times A \times Q$ is the transition relation of S. We denote $q \rightarrow q'$ if $(q, a, q') \in \rightarrow$, and $L : Q \rightarrow 2^{AP}$ where AP is a set of atomic propositions. $L(q)$ gives the set of propositions that hold at state q.

- ***Path in a Transition System:*** A path in a transition system is a sequence of states $\sigma = q_0, q_1, q_2, \ldots$ such that (q_i, q_{i+1}) for all $i > 0$.
- Let q be a state of S. The semantics of CTL is defined inductively as follows:
  - $\phi \in AP, q \models \phi \Leftrightarrow \phi \in L(q)$,
  - $q \models \phi \vee \psi \Leftrightarrow q \models \phi$ or $q \models \psi$,
  - $q \models \neg\phi \Leftrightarrow \neg\phi \in L(q)$,
  - $q \models AX\phi \Leftrightarrow \forall q'$ such that $(q, q') \in \rightarrow, q' \models \phi$,
  - $q \models EX\phi \Leftrightarrow \exists q'$ such that $(q, q') \in \rightarrow, q' \models \phi$,
  - $q \models AF\phi \Leftrightarrow \forall\sigma$ a path such that $\sigma(0) = q, \exists j$ such that $\sigma(j) \models \phi$,
  - $q \models EF\phi \Leftrightarrow \exists$ a path σ such that $\sigma(0) = q$ and $\exists j$ such that $\sigma(j) \models \phi$,
  - $q \models AG\phi \Leftrightarrow \forall\sigma$ a path such that $\sigma(0) = q, \forall j, \sigma(j) \models \phi$,
  - $q \models EG\phi \Leftrightarrow \exists\sigma$ such that $\sigma(0) = q$ and $\forall j, \sigma(j) \models \phi$,
  - $q \models A(\phi U \psi) \Leftrightarrow \forall\sigma$ such that $\sigma(0) = q, \exists j$ such that $\sigma(j) \models \psi$ and $\forall k$ such that $k < j, \sigma(k) \models \phi$,
  - $q \models E(\phi U \psi) \Leftrightarrow \exists$ a path σ such that $\sigma(0) = q, \exists j$ such that $\sigma(j) \models \psi$ and $\forall k$ such that $k < j, \sigma(k) \models \phi$.

3. **CTL's Extensions:**

   Temporal logic has gradually expanded, giving rise to other, much more expressive temporal logics such as Timed Computational Tree Logic (TCTL) [93] and Probabilistic Computational Tree Logic (PCTL) [94].

   (a) ***Timed Computation Tree Logic(TCTL)***

   TCTL [93] is an extension of CTL temporal logic that allows for the expression of properties involving temporal quantifications (e.g., properties like 'event i occurs in less than four seconds before another event j occurs'). The syntax of TCTL temporal logic is based on the formalism of CTL temporal logic, using propositions that involve constraints on clocks. Let ϕ and ψ denote two CTL formulas. TCTL formulas can be interpreted as follows:

   $\phi, \psi ::= p$: is an atomic proposition,

   $|\neg\phi| \vee \wedge\psi|\phi \vee \psi|\phi \Leftrightarrow \psi|\phi \Rightarrow \psi$: propositional logic operators,

   $|EF\phi|EG\phi|E\phi U\psi|A\phi U\psi|EX\phi|AF\phi|AG\phi|AX\phi$: CTL Operators,

   $|x \sim c|x - y \sim c$: x and y are clocks, $\sim \in \{<, >, \leq, \geq, =\}$, $c \in \mathbb{N}$.

   For example, $AG(\text{error} \Rightarrow AF(x \leq 3 \wedge \text{alarm}))$ is a TCTL formula that verifies the eventual triggering of an alarm within three time units in the case of an error.

   (b) ***Probabilistic Computation Tree Logic(PCTL)***

   PCTL [94] allows for specifying properties over discrete-time Markov chains. It was defined in 1994 by Hans Hansson and Bengt Jonsson, extending CTL temporal logic by adding the probabilistic operator P, along with quantitative extensions of the A and E operators.

   a. The syntax of a PCTL formula is defined as:

   - State formulas: $\phi := \text{true}|a|\phi \wedge \phi|\neg\phi|P_{\sim p}[\psi]$;
   - Path formulas: $\psi := X\phi|\phi U^{\leq k}\phi|\phi U \phi$ where:
   - a is an atomic proposition used to identify the states of interest for the analysis;

- $\sim \in \{<, >, \leq, \geq\}$ and $p \in [0, 1]$ is a probability;
- $k \in \mathbb{N}$;
- $X\phi$ means ϕ is true at the next step (next operator);
- $\phi_1 U^{\leq k} \phi_2$ means ϕ_2 is true in k steps and ϕ_1 is true up to that point (bounded until operator);
- $\phi_1 U \phi_2$ means ϕ_2 is eventually true and ϕ_1 is true up to that point (until operator).
- A property is always expressed using a state formula. Path formulas can only appear within the probabilistic operator P. The state formula $P_{\sim p}[\psi]$ means that (the path formula ψ is true with probability $\sim p$). The result of analyzing this qualitative property is either true or false. Furthermore, it is possible to evaluate the probability that the path formula ψ is true. The corresponding quantitative property is $P = [\psi]$. For example, in Figure 1.3, the probability formula for accessing state e_1 is: $P_{=}[e_1] = 0.99 + 0.01 * 0.95$.

2.2.3.1 Classifications of Temporal Properties

There are five major classes of properties that can be expressed in temporal logic [95]. In this section, we present these different classes of properties.

- ***Liveness Property:***

  A liveness property expresses that a state or event will necessarily occur in the future. This type of property can be expressed in LTL logic using the operator F (eventually) and the operator U (until). This property can also be expressed in CTL logic using the operator AF. Example: AFGOAL allows us to verify that the process will inevitably reach its target state GOAL.

- ***Safety Property:***

  A safety property expresses that something bad will never happen. This property can be expressed using the temporal operator G (always). Example: AGNOT FAIL allows us to verify that the process never enters a failure state.

- ***Reachability Property:***

  A reachability property expresses that a certain situation can be reached. This property is weaker than liveness, which expresses that a certain situation must necessarily be reached. The reachability property can be expressed using the temporal operator EF. To verify a reachability property, one simply needs to traverse the state graph and stop when such a state is reached or all states have been explored. Example: EFFAIL expresses that the process can fail.

- ***Fairness Property:***

  A fairness property expresses that something good will happen infinitely often. Fairness can be expressed using LTL temporal logic. However, it cannot be expressed using CTL temporal logic since CTL does not support A type operators (like FGp). Indeed, in CTL, the operators X, F, G, and U must be directly preceded by A or E.

- ***Deadlock Freeness:***

A deadlock freeness property expresses that a system does not get stuck in a state where it can no longer progress (no transition to another state is possible). A deadlock freeness property can be expressed by the CTL formula: $AG(EX\text{true})$. Another method to conclude the absence of deadlock is to verify that deadlock states are not part of the set of reachable states.

- ***Logical Properties with "Costs" and "Probability":***

  It is beneficial to verify properties that represent an extension of CTL with probabilistic and cost aspects. A logic called PWCTL is introduced [90]. This logic allows for expressing properties of the form:

  - $\mathbb{P}(\Diamond_{C\leq c}\varphi) \sim p$: comparing with a given probability p, the probability that in the future there will be a state where property φ is satisfied before the condition ($C \leq c$) is violated;
  - $\mathbb{P}(\Box C \leq c\varphi) \sim p$: comparing with a given probability p, the probability that in all states property φ is satisfied before the condition ($C \leq c$) is violated.

  The authors of [90] indicate that verifying the property: $A \models \mathbb{P}(\Diamond_{C\leq c}\varphi) \sim p$ amounts to verifying the property: $A \models \mathbb{P}_{A^*}(\bigcup_{\sigma\in\Sigma^*}\pi(s_0, \sigma o_\varphi)) \sim p$. The automaton A^* is obtained by modifying A by:

  - The invariant $C \leq c$ is conjoined with all invariants of the locations in A;
  - The arc $(l, \varphi, o_\varphi, \phi, l)$ is added to all locations l in A.

2.2.4 Formal Modeling Methods

Formal modeling refers to the creation of abstract models that describe the structure and behavior of a system using mathematical or logical formalisms. These models serve as a blueprint for understanding, analyzing, and verifying systems. Formal modeling methods are essential for system design, allowing engineers to reason about the system before its implementation.

2.2.4.1 Automata

Automata [65] are foundational models in formal methods, used to represent systems with discrete states and transitions. Automata are widely applied in the modeling of sequential and concurrent systems. They are often used in model checking and other verification techniques.

- Finite Automata (FA): FA represent systems with a finite number of states and are widely used in language recognition, parsing, and simple control systems.

- Timed Automata (TA): A variation of automata, timed automata incorporate clocks to model real-time systems where timing constraints are critical. Tools like UPPAAL are based on timed automata and are used for real-time verification.

- Probabilistic Automata: Used for systems with probabilistic behavior, these automata model systems that exhibit randomness in state transitions, such as in stochastic processes.

1. **Timed Automata (TA)**

   TA [96] [97] are classical automata equipped with non-negative real variables called clocks. Each automaton has a finite number of states, referred to as locations, and a finite set of transitions labeled by guards and clock-resetting actions. Each location is associated with a constraint on the clocks, called an invariant, which determines the duration of stay in a state before transitioning to the next one. In a state, a clock progresses continuously and synchronously with time. Its value represents the time elapsed since its last reset. A timed automaton allows for two types of transitions:

   - An action transition, which occurs when its guard is true.
   - A duration transition, which involves staying in the same state while the clocks progress continuously, respecting the invariant.

   As an illustration, consider the simple timed automaton in Figure 2.2, consisting of a clock x and two states e_0 and e_1. The automaton remains in state e_0 for up to one unit of time. When the clock x equals to 1, the automaton instantly takes the transition for which the guard is satisfied ($x == 1$) and resets the clock x. Since there is no invariant defined for state e_1, the automaton can stay there at least until the guard for its next destination is satisfied ($x \geq 3$). In other words, the automaton must remain in state e_1 for at least three units of time.

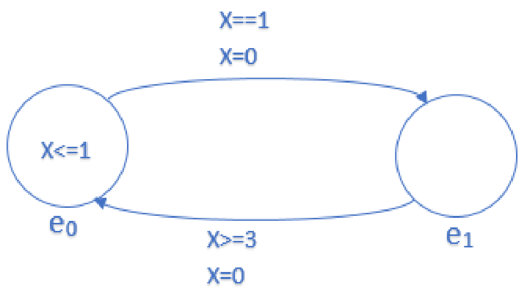


Figure 2.2: Example of a timed automaton.

   Formally, A timed automaton A is defined as:

$$A = (L, l_0, X, \Sigma, Inv, T)$$

   where:

   - L is a finite and non-empty set of states or locations,
   - $l_0 \in L$ is the initial location,
   - X is a finite set of clocks. Let $C(X)$ be the set of clock constraints called invariants, and 2^X the set of all subsets of X,
   - Σ is a finite alphabet defining a set of actions,
   - $Inv : L \rightarrow C(X)$ is a function that associates each state with a set of clock constraints of the form $x \sim c$ where $\sim \in \{<, \leq, =, \geq, >\}$, c is a constant, and $x \in X$,
   - $T \subseteq L \times C(X) \times \Sigma \times 2^X \times L$ is a finite set of transitions. Each element $t = (l, g, a, r, l') \in T$ represents a transition from l to l' such that g is the guard associated with t, which belongs to $C(X)$ and contains the set of constraints on the clocks, r is the set of clocks to be reset, and a is an action.

2. **Probabilistic Timed Automata (PTA):**

   Formally, a probabilistic timed automaton [98] is a tuple $(L, \bar{l}, \chi, \Sigma, inv, prob)$ where:

   - L: a finite set of locations (or states) and $\bar{l} \subseteq L$;
   - Σ: a set of events,
   - χ: a set of variables (called clocks). $\chi \in \mathbb{T}$ and $\mathbb{T} \in \{\mathbb{R}, \mathbb{N}\}$;
   - inv: $L \rightarrow Zone\{\chi\}$ is a function that associates each location with an invariant.
   - $Zone\{\chi\}$ is the set of logical expressions (guards) of the form: $x \sim c$, such that $x \in \chi$, $\sim \in \{\leq, =, \geq\}$, and $c \in \mathbb{N}$;
   - $prob$: a finite set of probabilistic transitions: $prob \subseteq L \times Zone(\chi) \times \Sigma \times Dist(2^{\chi} \times L)$. $Dist(2^{\chi} \times L)$ is the set of probabilistic distributions over all countable subsets of the product $2^{\chi} \times L$. In other words, each transition links one location to another, labeled by: an event, a guard, a set of variables to be reset, and finally, a transition probability. Thus, a probabilistic transition is a tuple (l, g, σ, p) where p is a probability function $p = \mu(X, l')$ defined for each pair $(X, l') \in 2^{\chi} \times L$; with $X \subseteq \chi$.

   (a) **Dynamics of PTA:**

   A state in a ProbTA is a pair $(l, v) \in L \times 2^{\chi}$ where l is a location and v is a valuation of the set of clocks in that location l. The valuation $v(l)$ must satisfy the invariant of l, denoted $v(l) \in inv(l)$. Initially, the system is in an initial state $(l_0, v(l_0))$ where all clocks are at 0, noted as $v(l_0) = \mathbf{0}$. From any state $(l, v(l))$, the system can evolve indeterministically in two possible ways:

   - Stay in the state $(l, v(l))$ as long as the elapsed time does not violate the guard: we always have $v(l) \in inv(l)$. This is called a "delay transition."
   - Or execute a discrete transition $t = (l, g, \sigma, p)$ allowing the system to move to another location l', with a probability $p = \mu(X, l')$ where X' is the set of clocks to be reset. This is denoted as $(l, v) \rightarrow^{t} (l', v')$. This transition is only possible if $v(l)$ satisfies the guard g and $v'(l')$ satisfies the invariant of l': $inv(l')$.

   (b) **Networks of PTA:**

   Let $PTA_i = (L_i, \bar{l}_i, \chi_i, \Sigma_i, inv_i, prob_i)$ be a ProbTA. The composition of two ProbTAs PTA_1 and PTA_2 is the ProbTA defined by $(L_1 \times L_2, (\bar{l_1}, \bar{l_2}), \chi_1 \cup \chi_2, \Sigma_1 \cup \Sigma_2, inv, prob)$, such that: $inv(l, l') = inv(l) \wedge inv(l')$ and $((l_1, l_2), g, \sigma, p) \in prob$ if and only if one of the following conditions is satisfied:

   - An evolution in the first automaton: $\sigma = \sigma_1 \in \Sigma_1$, and there is a transition $(l_1, g, \sigma, p_1) \in prob_1$ such that: $p = p_1 \otimes \mu(\emptyset, l_2)$. The transition probability equals the product of the probability of changing states in the first automaton and the probability of remaining in the same state in the second automaton;
   - An evolution in the second automaton: $\sigma = \sigma_2 \in \Sigma_2$, and there is a transition $(l_2, g, \sigma, p_2) \in prob_2$ such that: $p = p_2 \otimes \mu(\emptyset, l_1)$.
   - A synchronized evolution in both automata: $\sigma \in \Sigma_1 \cap \Sigma_2$, and there is a transition $(l_1, g_1, \sigma, p_1) \in \text{prob}_1$ and a transition $(l_2, g_2, \sigma, p_2) \in \text{prob}_2$, such that: $p = p_1 \otimes p_2$. The probability of transitioning is equal to the product of the probability of changing the state in the first automaton and the probability of changing the state in the second automaton.

2.2.4.2 Petri Nets (PNs)

Petri nets (PNs) [66] are a graphical and mathematical modeling tool used to represent concurrent and distributed systems. Petri nets excel at capturing the flow of information and control in systems with parallel processes. They are particularly useful in analyzing deadlock, synchronization, and resource allocation issues in complex systems. Petri Nets provide several benefits [99, 100, 101]:

- Strong mathematical foundation with intuitive graphical modeling, making complex systems easier to understand.
- Support for abstraction, refinement, and extensions (e.g., colored, timed, stochastic) to handle large, varied systems.
- Well-suited for qualitative and quantitative analysis with manageable state space representation.

However, they have certain limitations [99]:

- **State space explosion**: Increased complexity can rapidly expand the state space.
- **Modeling complexity**: Extensions can complicate modeling or affect verification.

Petri nets are widely used in both formal verification and modeling. Their graphical nature makes them intuitive for visualizing system dynamics, while their mathematical basis allows for rigorous analysis. A Petri net is a directed bipartite graph comprising places, transitions, and arcs that connect them.

1. **Petri Net Structure**

   A Petri net is defined as a 4-tuple $\mathcal{N} = \langle P, T, F, M_0 \rangle$ where

   - P is a finite, non-empty set of places,
   - T is a finite, non-empty set of transitions that does not overlap with P,
   - $F : (P \times T) \cup (T \times P) \longrightarrow \mathbb{N}$ is a flow relation representing the arcs,
   - $M_0 : P \longrightarrow \mathbb{N}$ represents the initial marking.

   The inputs (preset) and outputs (postset) for places and transitions are defined as follows:

   - The preset of place p, denoted by $\bullet p$, is $\bullet p = \{t \in T | F(t, p) > 0\}$.
   - The postset of place p, denoted by $p\bullet$, is $p\bullet = \{t \in T | F(p, t) > 0\}$.
   - The preset of transition t, denoted by $\bullet t$, is $\bullet t = \{p \in P | F(p, t) > 0\}$.
   - The postset of transition t, denoted by $t\bullet$, is $t\bullet = \{p \in P | F(t, p) > 0\}$.

2. **Dynamic Behavior of Petri Nets**

   The dynamic evolution of the marking in a Petri net governed by two rules: the "enabling rule" and the "firing rule." These rules rely on arc multiplicities and place markings.

   - **Enabling rule**: is concerned with the input arcs of a transition, while the firing rule considers both input and output arcs. A transition t is enabled at marking M, denoted by $M[t\rangle$, if $M(p) \geq F(p, t), \forall\, p \in \bullet t$.

- **Firing rule**: firing a transition represents the occurrence of its associated event. A transition is considered *enabled* if its pre-conditions are satisfied. Firing transition t that is enabled at marking M results in a new marking M', denoted by $M[t\rangle M'$, such that $M'(p) = M(p) + F(t, p)F(p, t),\ \forall\ p \in P$. Firing transition t removes tokens from each place in its preset according to the multiplicity of the arcs and adds tokens to each place in its postset in a similar manner.

In the following, fundamental definitions in Petri Net dynamics:

Definition 1 ***Transition Sequence****. A sequence of transitions, denoted as $\sigma = t_1, t_2, \ldots, t_n$, is considered fireable from an initial marking M_1 if there exists a sequence of markings $M_1, M_2, \ldots, M_n$ such that for each $i \in 1, 2, \ldots, n$, the marking M_i can transition to M_{i+1} by firing t_i.*

The notation $M_1[\sigma\rangle M_{n+1}$ indicates that the marking M_{n+1} can be reached from the initial marking M_1 by firing the transition sequence σ.

Definition 2 ***Reachability Set.*** *The reachability set for a Petri net with initial marking M_0, represented as $RS(M_0)$, is defined as the set of all markings M that can be reached from M_0 by some fireable sequence of transitions, i.e., $RS(M_0) = M \mid M_0[\sigma\rangle M$ and σ is a fireable sequence.*

Definition 3 ***Reachability Graph.*** *The reachability graph of a Petri net $\mathcal{N} = \langle P, T, F, M_0\rangle$, denoted as $RG(M_0) = \langle V, E\rangle$, is a directed graph where:*

i) *V represents the set of markings in the reachability set $RS(M_0)$,*

ii) *E is the set of directed edges (M_i, t, M_j), where $M_i, M_j \in RS(M_0)$ and $t \in T$ is a transition such that $M_i \xrightarrow{t} M_j$ is a valid firing step.*

In addition to its modeling capabilities, one of the main reasons for utilizing Petri nets is their analysis potential. Various techniques have been developed for the verification of Petri nets, allowing for the assessment of numerous properties associated with concurrent systems. These properties can be analyzed based on the reachability graph (behavioral properties) or independently of it (structural properties).

3. **Properties of Petri Nets**

Petri nets are widely used for their strong analysis capabilities, enabling the verification of key properties in concurrent systems. These properties can be examined through behavioral analysis via the reachability graph or through structural analysis independently.

- ***Reachability:*** The reachability problem for Petri nets involves determining if a marking M can be reached from the initial marking. This problem has been shown to be decidable.
- ***Liveness and Deadlock-freedom:*** A Petri net is considered live if every transition can be fired again in the future at any reachable marking. The liveness problem is decidable, as it is recursively equivalent to the reachability problem. A transition in a Petri net can be classified into various levels of liveness.

- ***Boundedness:*** A Petri net is bounded if its reachable set is finite. The boundedness property is also decidable. A Petri net is said to be *k-bounded* if any reachable marking contains at most k tokens in any place. A safe Petri net is 1-bounded.
- ***Home state and Reversibility:*** A marking is considered a home state if it can be reached from all other reachable markings. The problem of identifying a home state is decidable. A Petri net is reversible if its initial marking is a home state.
- ***Conservation:*** A Petri net is conservative if the number of tokens in any reachable marking remains constant, meaning that the total number of consumed tokens equals the total number of produced tokens.
- ***Persistence:*** A Petri net is persistent if the firing of one transition does not disable another transition when both are enabled at any reachable marking. This property is decidable.
- ***Fairness:*** Fairness can be categorized into bounded-fairness and unconditional fairness, with definitions and properties that can be analyzed through various techniques.

2.2.4.3 Reconfigurability-Based Petri Nets Extensions

Reconfigurable systems are evolving to become more complex and adaptable, designed to change their configurations during operation to better meet shifting needs. This dynamic reconfigurability must occur in real time, and research emphasizes its importance. Petri nets are a key tool in studying these systems, but traditional Petri nets struggle to represent systems with changing structures, making modeling and verification difficult. To address this, extensions to Petri nets that incorporate dynamic structures have been proposed. Additionally, rule-based graph transformations offer a graphical way to model reconfigurations. However, increasing a model's flexibility can reduce its ability to make decisions, creating a trade-off in the development of reconfigurable systems. The goal is to develop models that accurately reflect real-world adaptive systems, though this can complicate analysis, making some properties undecidable. Despite these challenges, research continues to advance in improving the analysis of reconfigurable systems.

2.2.4.4 Nested Petri Nets (NP-nets)

Nested Petri Nets (NP-nets) [102] extend colored Petri nets based on the "nets-within-nets" paradigm. Tokens in NP-nets are either atomic or net tokens, which are themselves marked Petri nets. The NP-nets formalism allows modeling hierarchical and multilevel systems, enabling interaction and synchronization between the system net and its nested nets. Formally, an NP-net is defined as:

$$\text{NP-net} = (N_1, \ldots, N_k, SN)$$

where:

- $N_1, \ldots, N_k$: *Element nets*, which are marked Petri nets representing the behavior of net tokens.
- SN: *System net*, a high-level Petri net managing the element nets.

Each place in the system net SN is typed by:

- A set S of atomic tokens, or
- A net token $M(N, S)$, which is a marked net derived from an element net N.

The system net is defined as:

$$SN = (P_{SN}, T_{SN}, F_{SN}, \nu, \rho, \Lambda)$$

where:

- P_{SN}: Set of places.
- T_{SN}: Set of transitions.
- F_{SN}: Set of arcs.
- ν: Typing function mapping each place to S or $M(N, S)$.
- ρ: Arc labeling function.
- Λ: Transition labeling function, including synchronization labels and silent transitions (τ).

The behavior of NP-nets is described by three types of steps:

- **Element-autonomous steps:** Transitions within a net token fire independently of the system net.
- **System-autonomous steps:** Transitions in the system net manipulate tokens without altering their internal markings.
- **Synchronization steps:** Simultaneous firing of a transition in the system net and associated transitions in the net tokens.

2.2.4.5 Reconfigurable Timed Net Condition/Event Systems (R-TNCES)

Reconfigurable Timed Net Condition/Event Systems (R-TNCES) [103] are an extension of Timed Net Condition/Event Systems (TNCES) [104] tailored to model and analyze reconfigurable discrete event control systems (RDECS). This formalism integrates dynamic reconfiguration mechanisms into the TNCES framework, enabling the system to adapt automatically to changes at runtime while maintaining coherence and correctness. In order to understand R-TNCES, we define, at first, TNCES.

1. **Timed Net Condition/Event Systems (TNCES):**

   A TNCES is a foundational framework for modeling modular, real-time systems. It extends Petri nets by incorporating condition and event signals for interaction between modules and adds time constraints for handling real-time behavior. A TNCES is formally defined as:

   $$\Gamma = (P, T, F, W, CN, EN, DC, V, Z_0)$$

   where:

   - P, T, F, W: Define the structure of the net (places, transitions, arcs, and weights).
   - CN, EN: Represent condition and event signals for inter-module communication.

- DC: Specifies time constraints for input arcs, ensuring real-time behavior.
- V: Defines the event-processing mode (AND/OR) for transitions.
- $Z_0 = (M_0, D_0)$: Represents the initial marking and clock positions.

While TNCES are effective for static systems, they lack inherent support for dynamic reconfiguration scenarios, such as runtime modifications of components, arcs, or condition/event signals.

2. **Reconfigurable Timed Net Condition/Event Systems (R-TNCES)**

R-TNCES extend TNCES to address the limitations by incorporating dynamic reconfiguration capabilities. The formalism allows modeling systems that can adapt at runtime in response to environmental changes, errors, or user requirements. An R-TNCES is defined as:

$$RTN = (B, R)$$

where:

- B: **Behavior Module**, representing the union of all possible configurations modeled as TNCES.
- R: **Control Module**, a set of reconfiguration functions enabling transitions between configurations.

The behavior module B is represented as:

$$B = (P, T, F, W, CN, EN, DC, V, Z_0)$$

with components similar to TNCES but encompassing multiple possible configurations. Each configuration is a valid TNCES.

- **Reconfiguration Functions:** Reconfiguration is managed by functions $r \in R$, each defined as:
$$r = (\text{Cond}, s, x)$$
  - Cond: A precondition specifying when the reconfiguration is applicable.
  - s: A structural modification instruction (e.g., adding/removing places, transitions, or signals).
  - x: A state-mapping function ensuring coherence between the states of configurations before and after reconfiguration.
- **Dynamic Behavior:**
  - Reconfiguration Events: When a reconfiguration function r is triggered, the current TNCES is updated to a new configuration.
  - State Coherence: The state-mapping function x ensures the transition between configurations does not disrupt the system's functionality.
  - Temporal Constraints: Time intervals and deadlines are preserved during reconfiguration, ensuring real-time behavior.

2.2.4.6 Self-Modifying Nets

Self-Modifying nets (SM-nets) were introduced in [105, 106] to extend the expressive capabilities of Petri nets by incorporating dynamic reconfiguration. These nets enhance the traditional Petri net framework by allowing the enabling and firing conditions of transitions to change based on the marking of places or specific arc weights, adding flexibility and complexity to their behavior.

In an SM-net, the weight of an arc can be influenced by the marking of a place, meaning that the number of tokens required or produced by firing a transition can vary dynamically. This introduces reconfigurability, where firing a transition can affect the enabling and firing conditions of subsequent transitions. For example, if the marking of a place becomes zero, the arc weight to a transition may become zero, making that transition a source transition (a transition with an empty preset).

A Self-Modifying net is formally defined as a quadruple $\mathcal{SM} = \langle P, T, F, M_0 \rangle$, where:

- P is a finite set of places,
- T is a finite set of transitions, disjoint from P,
- $F : (P \times P_1 \times T) \cup (T \times P_1 \times P) \longrightarrow \mathbb{N}$ is a flow relation that assigns weights (multiplicities) to arcs. Here, $P_1 = P \cup \{1\}$, and $1 \notin P$ represents a special identifier used in the flow relation. Specifically:
  - If $F(x, 1, z) = m$, it indicates that the arc from x to z has a weight m,
  - If $F(x, y, z) = n$, it means the weight of the arc from x to z is $n \cdot M(y)$, where $M(y)$ is the marking of place y.
- $M_0 : P \longrightarrow \mathbb{N}$ is the initial marking function, which assigns an initial number of tokens to each place.

2.2.4.7 Reconfigurable Object Nets (RONs)

Reconfigurable Object Nets (RONs), introduced in [67], are a type of high-level net that incorporates the concept of "nets and rules as tokens". They extend the "nets as tokens" paradigm, first presented in [107, 108, 109]. Thus, two distinct classes of tokens are defined: token-nets and token-rules. This formalism allows not only the manipulation of token markings but also the dynamic transformation of net structures during execution, which is particularly useful for reconfigurable systems, by distinguishing between two levels of nets: the system level and the token level.

- The system level consists of a high-level net and a rule system, where a place can contain either token-nets or token-rules. The marking at this level represents the distribution of nets and rules across different places. The firing behavior at the system level dictates how token-nets move between places and how the structures of token-nets change.
- At the token level, a token-net can fire independently, without undergoing movement or transformation, to represent a marking change within the token-level net. Alternatively, a token-net firing can be synchronized with the firing of a system transition, in which case the token-net is moved or transformed only if a token-net transition occurs.

The concept presented in [67] involves treating the modification of the net structure as a rule-based transformation of Petri nets, viewed through the lens of graph transformation systems. To gain a clearer understanding, we need to explore graph transformation systems and define transformation rules.

1. **Graph Transformation Systems (GTSs)**

   GTSs[110] provide a versatile and intuitive framework for modeling systems with evolving structures. At the core of a GTS lies an initial graph G_0, which represents the starting configuration of the system $\mathcal{G}$. Accompanying this graph is a set of transformation rules $\mathcal{R}$ that define how the structure changes over time. Each rule consists of a left-hand side (LHS) and a right-hand side (RHS), which describe how one configuration is transformed into another. Formally, A graph transformation system $\mathcal{G} = \langle G_0, \mathcal{R} \rangle$, where:

   - G_0 is an initial graph,
   - $\mathcal{R}$ is a set of transformation rules.

   A graph G is said to be generated by $\mathcal{G}$ if it is derived from G_0 through the sequential application of rules from $\mathcal{R}$. For a given graph G, which represents the current state of $\mathcal{G}$, a rule can be applied if there is a matching subgraph of G that corresponds to the LHS. This matching is determined via a graph morphism, which maps the LHS to a subgraph of G. When a match is found, the LHS is removed and replaced with the RHS, resulting in a new graph configuration. Some GTS frameworks also make use of an interface graph to indicate which parts of the structure must remain unchanged and how the RHS is integrated into the existing graph. In the following, we define the Petri nets morphisms and transformation rule.

2. **Petri Net Morphism**

   Given two Petri nets:

   $$PN_1 = (P_1, T_1, F_1, W_1, M_1)$$
   $$PN_2 = (P_2, T_2, F_2, W_2, M_2)$$

   A Petri net morphism $f : PN_1 \to PN_2$ is a pair of functions:

   $$f = (f_P, f_T)$$

   where:

   - $f_P : P_1 \to P_2$: A function mapping places in PN_1 to places in PN_2.
   - $f_T : T_1 \to T_2$: A function mapping transitions in PN_1 to transitions in PN_2.

   The functions f_P and f_T must satisfy the following:

   - ***Preservation of Flow Relation:***

     $$\forall (x, y) \in F_1, \quad (f(x), f(y)) \in F_2$$

     where $x, y \in P_1 \cup T_1$, and f is extended to the union of places and transitions.
   - ***Preservation of Weights:***

     $$\forall (x, y) \in F_1, \quad W_1(x, y) = W_2(f(x), f(y))$$

     ensuring that the weights of arcs between corresponding elements are preserved.

- ***Preservation of Initial Marking:***

  $$\forall p \in P_1, \quad M_2(f_P(p)) = M_1(p)$$

  meaning that the initial marking of places in PN_1 is mapped to the corresponding marking in PN_2.

3. **Transformation Rule**

   In Double-Pushout approach for Petri nets, A transformation rule provides a formal mechanism for modifying a Petri net's structure by replacing specific sub-nets with new ones while preserving consistency and correctness. The rule defines the conditions under which a transformation can be applied, the parts of the net to be removed, and the parts to be added. A transformation rule is formally defined as:

   $$r = (L \xleftarrow{i_1} I \xrightarrow{i_2} R)$$

   where:

   - L: The **left-hand side** net, representing the part of the net to be replaced.
   - I: The **interface net**, a shared substructure that ensures consistency during the transformation.
   - R: The **right-hand side** net, specifying the new structure to replace L.
   - $i_1 : I \to L$: A strict morphism mapping the interface I to the left-hand side L.
   - $i_2 : I \to R$: A strict morphism mapping the interface I to the right-hand side R.

   The application of a transformation rule is based on two graph gluing constructions, commonly referred to as pushouts. These constructions ensure consistency and correctness in the transformation process. Applying a rule r to a Petri net G involves the following steps:

   - Match identification by locating a sub-net in G that matches L via a morphism $m : L \to G$.
   - Obsolete and fresh elements identification by identifying elements in L not in the image of i_1 (obsolete elements to be removed) and elements in R not in the image of i_2 (fresh elements to be added).
   - First pushout by constructing the context net C by removing obsolete elements of L from G along the interface I, ensuring that $G = C +_I L$.
   - Second pushout by gluing C and R along I, resulting in the transformed Petri net G', such that $G' = C +_I R$.

   The transformation is denoted by: $G \xrightarrow{r,m} G'$ where G' is the resulting Petri net. The resulting Petri net $G' = \langle P_{G'}, T_{G'}, F_{G'}, M^0_{G'} \rangle$ is defined as:

   $$P_{G'} = P_C \uplus (P_R \setminus P_I)$$
   $$T_{G'} = T_C \uplus (T_R \setminus T_I)$$

where $\uplus$ represents disjoint union, and $\setminus$ represents set difference. The flow and marking functions are given by:

$$F_{G'}(v,w) = \begin{cases} F_C(v,w) & \text{if } v,w \in P_C \cup T_C \\ F_R(v,w) & \text{if } v,w \in P_R \cup T_R \\ 0 & \text{otherwise} \end{cases}$$

$$M^0_{G'}(p) = \begin{cases} M^0_C(p) & \text{if } p \in P_C \setminus P_R \\ M^0_R(p) & \text{if } p \in P_R \setminus P_C \\ M^0_C(p) + M^0_R(p) - M^0_I(p) & \text{otherwise} \end{cases}$$

2.2.4.8 Net Rewriting Systems (NRSs) and its Extensions

NRSs [111, 112] was introduced to enhance the capabilities of standard Petri net (PN) frameworks. They represents a combination Petri nets elements with graph transformation systems. In the following, the NRSs and its extensions are presented.

1. **Net Rewriting Systems (NRSs)**

   An NRS is defined as $\mathcal{N} = \langle G_0, M_0, \mathcal{R} \rangle$, where:

   - G_0 is the *initial unmarked Petri net.*
   - M_0 is the *initial marking.*
   - $\mathcal{R}$ is a set of *rewriting rules.*

   Each rewriting rule $r = \langle L, R, I, O \rangle$ consists of:

   - L: the *left-hand side (LHS)* Petri net (unmarked),
   - R: the *right-hand side (RHS)* Petri net (unmarked),
   - $I \subseteq (P_L \times P_R) \cup (T_L \times T_R)$: the *input interface*,
   - $O \subseteq (P_L \times P_R) \cup (T_L \times T_R)$: the *output interface.*

   Applying a Rewriting Rule is as follows:

   - ***Matching:*** A *morphism* m is used to find a match $\mathcal{M}$ for the LHS L in the original Petri net G, ignoring markings.
   - ***Context Graph:*** The part of G not matched by m forms the *context graph*, ensuring that the input and output interface conditions are met.
   - ***Rewriting:*** After matching, the LHS part of G is replaced by the RHS, creating a new Petri net $H = \langle P_H, T_H, F_H, M^0_H \rangle$.
     - The *marking* M^0_H is computed by summing the markings of places in G that correspond to places in R:

     $$M^0_H(p) = \sum_{p' \in \pi(p)} M(m(p'))$$

     where $\pi(p) = \{p' \mid (p', p) \in I \cup O\}$.

- The *flow function* F_H is updated based on the interface relations:

$$F_H(v, w) = \sum_{w' \in \pi_I(w)} F_G(v, m(w'))$$

for transitions between places and transitions in the new net.

2. **Improved Net Rewriting Systems (INRSs)**

INRSs [113, 114, 115, 116, 117] extend NRSs by allowing the reconfiguration of live, bounded, and reversible (LBR) Petri nets (PNs) while preserving these properties, eliminating the need for re-verification. INRSs consist of three key components:

- NRSs as the foundational framework,
- Well-Behaved Nets (WBNs) to define interface relations between the left and right sides of rules,
- A Net Block Type Library that specifies allowable subnet types for rule components.

INRSs enable the replacement of subnets while preserving LBR properties. The system applies rules to subnets from the defined library (e.g., state machines, marked graphs).

An INRS is defined as $\mathcal{N} = \langle G_0, \mathcal{R}, \mathcal{L} \rangle$, where:

- G_0 is the initial PN configuration,
- $\mathcal{R}$ is a set of rewriting rules,
- $\mathcal{L}$ is the net block library.

A rule is a 4-tuple $r = \langle L, R, I, O \rangle$, where:

- L and R are the left and right sides of the rule (PNs from $\mathcal{L}$),
- I and O are the input and output interface relations (sets of places or transitions).

Rules apply if, in addition to the NRS conditions, the context graph satisfies the preset and postset conditions for the input and output interfaces.

3. **Reconfigurable Petri Nets (RPNs)**

RPNs [118, 111, 112] extend traditional Petri Nets (PNs) by enabling dynamic reconfiguration through rewriting rules that modify flow relations, while keeping places and transitions unchanged. RPNs combine features of Petri Nets and Self-Modifiying Nets (SM-nets).

An RPN is defined as $N = \langle P, T, \mathcal{R}, \gamma_0 \rangle$, where:

- P and T are sets of places and transitions,
- $\mathcal{R}$ is a set of rewriting rules,
- γ_0 is the initial configuration.

A rule $r = \langle D, ^\bullet r, r^\bullet, \mathbb{C}, \mathbb{M} \rangle$ modifies flow relations in D and is applicable if the current markings meet preconditions.

RPNs can be transformed into equivalent Petri nets, which can be analyzed using standard Petri net analysis methods like reachability graphs.

Example: The equivalent Petri net includes cloned places, configuration-specific transitions, and rule-based transitions, enabling traditional PN analysis techniques.

2.3 Introduction to Genetic Algorithms

Genetic Algorithms (GAs) are a class of optimization and search techniques inspired by the principles of natural selection and genetics [119]. They belong to the broader field of evolutionary algorithms, which also includes methods like genetic programming, evolution strategies, and differential evolution. GAs use mechanisms inspired by biological evolution, such as selection, crossover (recombination), and mutation, to evolve solutions to complex problems. These algorithms are particularly effective in solving optimization problems that are high-dimensional, non-linear, and poorly defined or lack a smooth search space.

To understand genetic algorithms, it's important to grasp key concepts which are: (1) ***Population:*** A subset of all possible solutions to a problem, maintained within the search space. (2) ***Chromosome (Individual):*** A solution represented as a finite-length vector of components called ***genes***, Figure 2.3 illustrates the population contents, with each gene being a variable part of the solution. (3) ***Fitness Function:*** A measure that assigns a fitness score to each individual, reflecting its ability to solve the problem and compete with others. (4) ***Genetic Operators:*** Methods that alter the genetic makeup of offspring, improving their performance over their parents. Together, these elements help drive the evolutionary process in genetic algorithms.

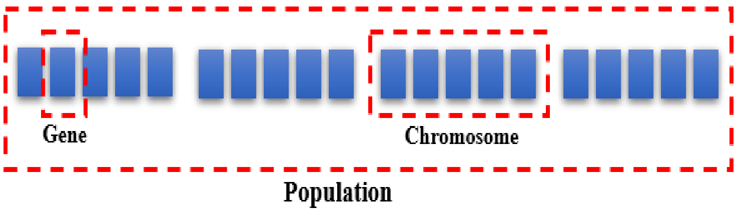


Figure 2.3: Population, chromosome and gene.

GAs are powerful tools for optimization, especially in cases where the search space is large, complex, and poorly understood. Unlike traditional optimization methods that may require differentiability or continuity, GAs can handle discrete, non-continuous, and noisy functions. They are particularly useful in:

- Multi-modal optimization: GAs are effective in exploring multiple peaks or optima in a fitness landscape, helping find global optima in situations where traditional optimization methods struggle with local optima.
- Large, complex search spaces: GAs can efficiently explore large solution spaces without requiring an exhaustive search or domain-specific knowledge, making them ideal for high-dimensional problems.
- Non-differentiable objective functions: Since GAs do not rely on gradient-based methods, they can be applied to problems where the objective function is not smooth or continuous.

2.3.1 Fundamentals of Genetic Algorithms

A genetic algorithm operates by simulating the process of natural selection. The core idea is to represent potential solutions to a problem as individuals in a population. These individuals undergo processes of reproduction and survival, guided by a fitness function that measures their performance in solving the problem. GAs proceed through generations, with each generation evolving to improve the population's quality. Key components of a GA include [120]:

- **Representation (Encoding):** Solutions are encoded as chromosomes (strings or arrays), where each chromosome represents a candidate solution. The choice of encoding depends on the problem domain, with common schemes including binary, octal, hexadecimal, permutation, value-based, and tree encoding.

  - Binary encoding is widely used, where genes are represented as strings of 1s and 0s. It allows fast crossover and mutation but requires effort in conversion and may struggle with certain engineering problems.
  - Octal and hexadecimal encoding use octal (0–7) and hexadecimal (0–9, A-F) numbers to represent genes, respectively.
  - Permutation encoding is used for ordering problems, where genes are represented by sequences of numbers.
  - Value-based encoding represents genes as strings of real, integer, or character values and is useful for complex problems, especially in neural networks.
  - Tree encoding represents genes as trees of functions or commands, often used in evolving programs or expressions.

  Each encoding scheme has its advantages depending on the problem being solved.

- **Population Initialization:** A population of candidate solutions is randomly generated, though initial populations can also be seeded to cover specific regions of the solution space, depending on the problem.

- **Selection:** Selection determines which individuals will reproduce and form the next generation. It is a key step in GAs that determines which individuals will reproduce. The convergence rate of a GA depends on the selection pressure. Common selection techniques include roulette wheel, rank, tournament, Boltzmann, and stochastic universal sampling.

  - Roulette wheel selection assigns a portion of the wheel to each individual based on their fitness and randomly selects solutions. However, it can suffer from errors due to its stochastic nature. Modifications like deterministic roulette wheel selection reduce these errors.
  - Rank selection ranks individuals based on fitness, giving each individual a chance according to their rank, which helps avoid premature convergence.
  - Tournament selection involves selecting individuals in pairs based on fitness, with the higher fitness individual advancing to the next generation.
  - Stochastic universal sampling (SUS) is a variation of roulette wheel selection, using a random starting point and selecting individuals at evenly spaced intervals, offering equal chances to all individuals.
  - Boltzmann selection uses entropy and sampling methods to address premature convergence, with a higher probability of selecting the best individual but a risk of information loss, which can be mitigated by elitism.
  - Elitism ensures that the best individual always moves to the next generation, improving performance by retaining high-quality solutions.

- **Crossover (Recombination):** Crossover is the process of combining parts of two parent chromosomes to produce offspring. This simulates genetic recombination in biological systems and allows the GA to explore new solutions by mixing traits from both parents. Common crossover techniques include single-point, two-point, k-point, uniform, partially matched (PMX), order (OX), precedence-preserving (PPX), shuffle, reduced surrogate (RCX), and cycle crossover.

  – Single-point crossover randomly selects a crossover point and swaps genetic material beyond this point between two parents, generating offspring from the swapped parts.
  – Two-point and k-point crossover [121] select multiple random crossover points, exchanging genetic material between parents based on these points.
  – Uniform crossover treats each gene independently, randomly deciding whether to swap each gene between parents.
  – Partially matched crossover (PMX) [122] is a frequently used operator, where part of one parent's genetic material is swapped with the corresponding part of the other parent. The remaining genes are copied from the second parent.
  – Order crossover (OX) [123] copies parts of one parent to the offspring and fills the remaining spaces with genes from the second parent that aren't already included, useful for ordering problems.
  – Precedence-preserving crossover (PPX) [124] maintains the ordering of genes from the parent solutions, ensuring certain constraints are preserved in the offspring.
  – Shuffle crossover [125] randomizes the genes before crossover and restores their original order after, minimizing bias from the crossover point.
  – Reduced surrogate crossover (RCX) [126] reduces unnecessary crossovers by preventing swapping when parents have identical genetic sequences.
  – Cycle crossover [127] generates offspring by taking genes from alternating cycles of parents, ensuring that all positions are filled by referencing their positions in the parents.

  In terms of performance, single-point and k-point crossover are easy to implement, while uniform crossover is suited for large gene subsets. Order and cycle crossovers offer better exploration, and PMX is particularly effective. However, RCX and cycle crossover can suffer from premature convergence.

- **Mutation:** Mutation is a key operator in genetic algorithms that preserves genetic diversity between populations. Common mutation operators include displacement, simple inversion, and scramble mutation [128].

  – Displacement mutation (DM) moves a substring within an individual solution to a random position, ensuring the new solution remains valid. Variants of DM include exchange mutation, which swaps parts of the solution, and insertion mutation, which inserts a part of the solution at a different location.
  – Simple inversion mutation (SIM) reverses a substring between two specified points within the solution.
  – Scramble mutation (SM) rearranges elements within a specific range of the solution randomly, checking if the new solution improves the fitness value.

The process of the genetic algorithm is depicted in Figure 2.4

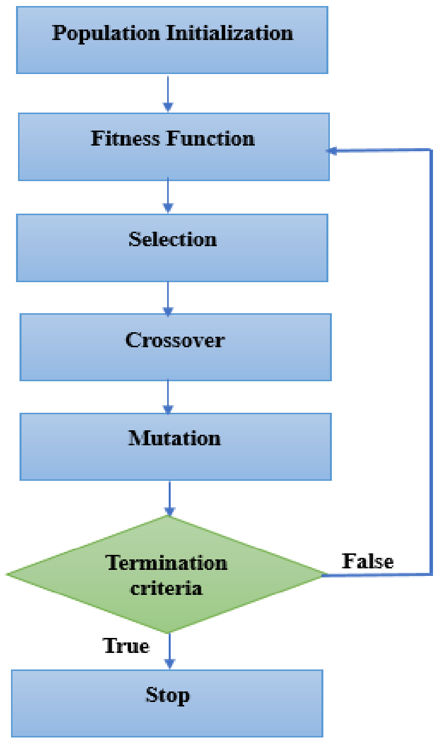


Figure 2.4: Workflow of the genetic algorithm.

2.3.2 Challenges in Applying GAs to RSs

While GAs have shown great promise in optimizing reconfigurable systems, several challenges must be addressed:

- Computational Complexity: GAs can be computationally expensive, particularly when dealing with large populations or complex systems with many variables. The need to evaluate fitness for each individual in the population across multiple generations can lead to high computation costs, especially in real-time applications.

- Premature Convergence: GAs are susceptible to premature convergence, where the population becomes too similar, leading to a loss of genetic diversity and stagnation in the search process. To combat this, techniques such as maintaining diversity through mutation, using adaptive mutation rates, or employing multi-objective optimization are often applied.

- Scalability: As reconfigurable systems grow in complexity (e.g., more parameters, larger system sizes), the search space increases exponentially, making the optimization problem harder. Hybrid approaches that combine GAs with other techniques can help improve scalability and efficiency.

- Real-Time Constraints: In certain reconfigurable systems, such as robotics or autonomous vehicles, real-time optimization is required. GAs can be slow to converge, which may be a disadvantage in time-sensitive applications. Parallelism, multi-threading, or more efficient genetic operators are some ways to speed up the algorithm.

Despite these challenges, the flexibility and robustness of GAs make them an attractive option for reconfigurable systems optimization. Advances in parallel computing, multi-objective optimization, and hybrid algorithms are expected to further improve their performance and applicability.

2.3.3 GAs Variants

GAs have been modified into several variants to enhance their performance across different problem types. These variants are generally grouped into five categories: real and binary coded GAs, multi-objective GAs, parallel GAs, chaotic GAs, and hybrid GAs. Overall, these GA variants address specific challenges and improve performance in diverse applications, such as multi-objective optimization, parallel computing, and chaotic problem-solving, etc.

1. **Binary Coded GAs:**

   These represent solutions using binary strings. Although effective for specific tasks, binary GAs face challenges like difficulty achieving precision.

2. **Real-Coded GAs:**

   Used for problems where the solution naturally involves real values. They are robust and efficient but prone to premature convergence, requiring modifications to genetic operators to improve performance.

3. **Multi-objective GAs (MOGAs):**

   Multi-objective Genetic Algorithm (MOGAs) is an enhanced version of the standard Genetic Algorithm (GA), differing primarily in how fitness functions are assigned. While the core steps remain similar to GA, MOGA aims to generate an optimal Pareto Front in the objective space, ensuring that no fitness function can be improved without negatively impacting others. The key goals of MOGA are to achieve convergence, maintain diversity, and ensure broad coverage of solutions.

4. **Parallel GAs:**

   Parallel GAs aim to improve computational efficiency by distributing the work across multiple processors. They include:

   (i) Master-Slave Parallel GAs: Fitness evaluation is distributed among several processors, but these systems may have high computational time. (ii) Fine-Grained Parallel GAs: Involve more localized interactions among individuals, used for real-life problems. (iii) Coarse-Grained Parallel GAs: Individuals exchange solutions among sub-populations, and the system may involve complex management of resources like GPUs.

5. **Chaotic GAs:**

   Chaotic systems are integrated into GAs to address premature convergence. Chaotic maps replace traditional randomization in genetic operators, enhancing solution accuracy. However, they tend to suffer from high computational complexity.

6. **Hybrid GAs:**

   Hybrid GAs combine genetic algorithms with other optimization techniques to improve solution quality, efficiency, and feasibility, especially for complex problems. This variant's algorithms enhance the search capability by improve the solution quality, and are used to fine-tune the parameters of the genetic algorithm.

2.3.4 Multi-Objective Genetic Algorithms (MOGAs)

MOGAs [129] are a class of genetic algorithms designed to solve optimization problems with multiple, often conflicting objectives. MOGAs are applied in various domains, such as: (i) Reliability optimization: Balancing reliability and cost in systems like nuclear power plants. (ii) Resource allocation: Optimizing time, cost, and resources in manufacturing systems. MOGAs have advantages, such as robustness in handling non-convex, multi-modal problems, simultaneous generation of diverse trade-off solutions and eliminating the need for predefined weights or scaling. However, these algorithms have limitations, such as, computationally intensive due to large populations and multiple objectives, performance degrades with an increasing number of objectives (scalability issues), and hyperparameter sensitivity requires careful tuning [130].

2.3.4.1 Introduction to Multi-objective Optimization

Multi-objective optimization involves solving problems with multiple conflicting objectives. Unlike single-objective optimization, which seeks a single optimal solution, multiobjective optimization aims to find a set of trade-off solutions that balance the objectives. The solutions are evaluated based on *Pareto optimality*, where no improvement in one objective is possible without sacrificing performance in another [131].

A multi-objective optimization problem can be formally defined as [132]:

$$\text{Minimize } z(x) = \{z_1(x), z_2(x), \dots, z_K(x)\}, \quad x \in X$$

where:

- $z(x)$ is the vector of K objective functions.
- $x = \{x_1, x_2, \dots, x_n\}$ is the decision variable vector.
- X is the feasible solution space, defined by constraints such as $g_j(x) \leq b_j$.

A solution $x^* \in X$ is *Pareto optimal* if no other solution $x \in X$ dominates x^*. For two solutions $x, y \in X$, x dominates y ($x \prec y$) if:

$$z_i(x) \leq z_i(y) \quad \forall i \in \{1, 2, \dots, K\}, \quad \text{and} \quad z_j(x) < z_j(y) \quad \text{for at least one } j.$$

The set of all non-dominated solutions forms the *Pareto front*, representing the trade-offs among objectives. Multi-objective genetic algorithms (MOGAs) aim to approximate this Pareto front.

2.3.4.2 MOGAs Fundamentals

MOGAs aim to generate a set of diverse, non-dominated solutions, approximating the Pareto front. The following covers the key components of MOGAs [131].

1. **Pareto Dominance and Optimization Goals**

   Pareto dominance is central to MOGAs. Solutions are evaluated based on whether they are dominated by others in the population. The main goals of MOGAs are:

   (a) **Convergence:** Ensure that the solutions are as close as possible to the true Pareto front.

   (b) **Diversity:** Maintain a uniform distribution of solutions along the Pareto front.

(c) **Completeness:** Capture the entire spectrum of trade-offs, including extreme points.

2. **Fitness Assignment**

   Fitness assignment in MOGAs is typically based on Pareto ranking, which divides the population into *non-dominated fronts*. Another approach is the *weighted sum method*, which converts multiple objectives into a single scalar function:

   $$\text{Minimize } Z(x) = \sum_{i=1}^{K} w_i z_i(x)$$

   where:

   - w_i are weights for each objective, with $\sum_{i=1}^{K} w_i = 1$.
   - $Z(x)$ is the scalarized objective function.

   While simple, this method struggles with non-convex Pareto fronts and requires careful selection of weights.

3. **Diversity Preservation**

   Diversity is crucial for ensuring a uniform spread of solutions across the Pareto front. Techniques include:

   - ***Fitness Sharing*** Reduces the fitness of solutions in densely populated regions. The niche count for a solution x_i is calculated as:

     $$nc(x_i) = \sum_{j=1}^{N} \max\left(0, 1 - \frac{d(x_i, x_j)}{\sigma_{\text{share}}}\right)$$

     where:

     – $d(x_i, x_j)$ is the distance between solutions x_i and x_j in the objective space.
     – σ_{share} is the niche size.

     Fitness is adjusted as:

     $$f'(x_i) = \frac{f(x_i)}{nc(x_i)}.$$

   - ***Crowding Distance*** Measures the sparsity of solutions around a candidate x_i:

     $$cd(x_i) = \sum_{k=1}^{K} \frac{z_k(x_{i+1}) - z_k(x_{i-1})}{z_k^{\max} - z_k^{\min}}$$

     where:

     – x_{i+1} and x_{i-1} are adjacent solutions in the sorted list for the k-th objective.
     – $z_k^{\max}$ and $z_k^{\min}$ are the maximum and minimum values of the k-th objective.

4. **Elitism**

   Elitism ensures the preservation of high-quality solutions across generations. Two common strategies are:

   - Retaining non-dominated solutions directly within the population.
   - Maintaining an external archive of non-dominated solutions.

2.3.4.3 Prominent Multi-objective Genetic Algorithms

Several MOGAs have been proposed. In the following, brief descriptions for the popular multiobjective genetic algorithms are presented:

1. **Vector Evaluated Genetic Algorithm (VEGA)**

   VEGA [133] is the first MOGA designed to approximate the Pareto front. This algorithm divides the population into subgroups, each optimized for a single objective. Solutions from these subgroups are recombined through crossover and mutation. However, while simple to implement, VEGA often produces solutions biased toward extreme objectives, failing to achieve a well-distributed Pareto front.

2. **Niched Pareto Genetic Algorithm (NPGA)**

   NPGA [134] employs a tournament selection mechanism based on Pareto dominance. As innovations, it uses a niching strategy to maintain diversity and employs dominance-based comparisons instead of aggregating objectives. This algorithm ensures better exploration of the solution space and avoids convergence to a specific region of the Pareto front.

3. **Nondominated Sorting Genetic Algorithm (NSGA)**

   NSGA [135] is considered as one of the first Pareto-based evolutionary algorithms. Its key Features: (i) Ranks the population based on Pareto dominance. (ii) Assigns fitness inversely proportional to rank, favoring non-dominated solutions. However, as a drawback, it suffers from computational complexity due to sorting, especially for large populations.

4. **Fast Nondominated Sorting Genetic Algorithm (NSGA-II)**

   An enhancement of NSGA, NSGA-II [136] addresses its computational limitations. The algorithm's innovations: (i) Fast Sorting: Reduces complexity to, where is the number of objectives and is the population size. (ii) Crowding Distance: Maintains diversity without introducing additional parameters. (iii) Elitism: Ensures preservation of the best solutions across generations. NSGA-II remains one of the most widely used MOGAs.

5. **Strength Pareto Evolutionary Algorithm (SPEA)**

   SPEA [137] incorporates an external archive to store non-dominated solutions. The algorithm's core Components: (i) Strength Value: Solutions are assigned fitness based on the number of dominated individuals. (ii) Archive Maintenance: Prunes the archive to ensure diversity and manageable size. This algorithm effectively balances exploration and exploitation.

6. **Pareto Archived Evolution Strategy (PAES)**

   PAES [138] is considered as the simple yet powerful algorithm. It is based on evolution strategies. PAES maintains a single candidate solution and an external archive of non-dominated solutions. It uses a grid-based mechanism to ensure diversity in the archive. It, particularly, suited for problems with low-dimensional objectives.

7. **Random Weighted Genetic Algorithm (RWGA)**

   RWGA [139] generates random weight vectors for each solution during the selection phase, allowing simultaneous optimization in multiple directions.RWGA doesn't need for predefining weight vectors. However, it performs poorly on non-convex Pareto fronts.

8. **Multi-objective Genetic Algorithm (MOGA)**

   This approach is developed to address the limitations of VEGA. MOGA [129] introduces Pareto-based ranking and a niching strategy to maintain diversity. It produces a more uniform distribution of solutions.

9. **Rank-Density Based Genetic Algorithm (RDGA)**

   RDGA [140] incorporates rank and density information to guide the search. RDGA penalizes regions with high population density to encourage exploration of sparsely populated areas for improving coverage of the Pareto front.

2.4 Conclusion

This chapter has provided a comprehensive overview of two key areas fundamental to this thesis: formal methods and genetic algorithms. In the first section, we explored formal methods as a rigorous foundation for the analysis and verification of complex systems. We discussed formal verification techniques, with an emphasis on model checking, as well as property specification formalisms, highlighting the role of temporal logics such as LTL and CTL, and their extensions. The section also addressed formal modeling methods, with a particular focus on the extensions of automata and Petri nets that support reconfigurability, which are crucial for analyzing reconfigurable systems.

In the second section, we delved into genetic algorithms, covering their definition, fundamentals, and variants, with a specific focus on multi-objective genetic algorithms. We also analyzed the challenges of applying genetic algorithms in the context of reconfigurable systems, laying the groundwork for their application in optimization problems.

The insights from this chapter serve as a foundation for the contributions outlined in the subsequent chapters.

The next chapter will introduce the first contribution of this thesis: the use of model checking and probabilistic automata for the analysis of reconfigurability in Mobile wireless sensor networks and Internet of Things. By leveraging formal verification methods, this contribution aims to ensure the correctness and adaptability of reconfigurable networks.

Following this, the thesis will present a second major contribution focusing on the application of high-level Petri nets RONs (Reconfigurable Object Nets) and genetic algorithms to propose new formalism that can model, analysis and optimize reconfigurable manufacturing systems. These contributions demonstrate the synergy between formal methods and genetic algorithms in tackling complex and reconfigurable systems.

Chapter 3

Model Checking for Formal Modeling and Verification of Reconfigurable Systems: Protocols Performance Evaluation

3.1 Introduction

Reconfigurability is a fundamental property in modern systems, enabling them to adapt and evolve based on changing operational conditions or requirements. While Chapter 1 provided a broad overview of reconfigurable systems, this chapter focuses on the application of formal methods in verifying and evaluating the performance of reconfigurable communication protocols. Specifically, it highlights our two key contributions [141], [142] that explore how formal modeling and verification can be applied to reconfigurable systems.

The chapter is organized as follows. Section 3.2 presents the first contribution, which involves the formal modeling and performance evaluation of the CFMA MAC protocol used in mobile wireless sensor networks. In this section, we use formal methods, including probabilistic timed automata and the UPPAAL SMC tool, to assess both the correctness and performance of the protocol, ensuring its efficiency and reliability in a mobile wireless sensor network environment.

Section 3.3 introduces the second contribution, which focuses on applying similar formal methods to the MQTT 3.1.1 protocol, a widely-used communication protocol in the Internet of Things (IoT). This section extends the approach from Section 1 to study the protocol's behavior and performance in IoT systems, which are highly dynamic and require adaptability to varying network conditions. Formal verification of the MQTT protocol in such environments helps to ensure that it meets both qualitative and quantitative performance expectations.

Section 3.4 concludes the chapter by summarizing the key insights from the two contributions and reflecting on the implications of these findings for reconfigurable systems in communication protocols.

3.1.1 Context of the Work

Wireless Sensor Networks (WSNs) and Internet of Things (IoT) systems are foundational technologies enabling pervasive connectivity in various domains. With the increasing deployment of mobility-based WSNs and dynamic IoT networks, the challenges of ensuring reliable communication, optimal energy utilization, and robust performance under changing topologies are paramount. Existing studies on WSNs and IoT protocols, particularly in static environments, have leveraged formal methods to ensure compliance with standards and verify properties such as liveness, safety, and reachability ([143, 144, 145, 146]). However, the formal analysis of mobile WSNs remains limited, with most performance evaluations relying on simulation such as in ([147, 148, 149, 150]).

Similarly, while IoT protocols have been extensively simulated for performance metrics ([151, 152, 153, 154]), formal methods have only been applied sporadically. Notable works ([155, 156, 157]) have focused on testing IoT protocols like MQTT and CoAP, yet a comprehensive formal study addressing mobility dynamics and essential networking properties remains scarce.

The formal verification has been used in networking domain to verify communication protocols in both levels: network layer ([158], [159], [160]) and MAC layer ([143], [144], [145], [161], [162], [163], [164]). In [143], the authors proposed protocols dedicated to WSNs using Model Driven Software Engineering (MDSE). They used CPN-tool (coloured Petri Nets) [165] to perform the formal study. The authors in [144] proposed an extension of Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) protocol where the Distributed Coordination Function (DCF) mode was introduced. They used UPPAAL tool for the formal modeling with the timed automata and a qualitative verification of some properties was accomplished.

The latter work [144] was extended by [145] where the quantitative verification of the aforementioned protocol was considered. In [145], the authors used UPPAAL SMC tool for making a performance analysis using the statistical model checking. In the work reported in [161], the authors studied the protocol IEEE 802.11 CSMA/CA using Markov chain. Likewise, the aforementioned protocol were studied in [162] where the authors used also Markov chain to model the exponential backoff process. On another hand, the authors in [166] took a sub-protocol of the standard IEEE 802.11 called two-way handshake and used both the UPPAAL and PRISM tools for the qualitative and quantitative verification. Another WSNs protocol was studied in [163] which is Timing-sync protocol [167] where the authors used stochastic timed automata for the modeling and UPPAAL tool for the verification of qualitative properties. A comparative study was made in [164] between IEEE 802.11 protocol for wireless ad-hoc and Sensor MAC (S-MAC) using PRISM and Probabilistic Computation Tree Logic (PCTL) to define the properties.

All the aforementioned works used formal methods in the case of static nodes. Indeed, few works investigated the formal study of mobile WSNs. The majority of these works used extensive simulation for their performance evaluation in their studies as reported in ([147], [148], [149], [150]). Among the few works tackling the formal study of mobile WSNs protocols, the two works reported in [168] and [169]. In [168], the authors presented a formal approach for the specification, verification and simulation of wireless sensor network with mobile nodes using Prototype Verification System (PVS) tool. The formal study of the mobility was also addressed in [169] where the authors proposed an approach for the formal modeling and verification of ad-hoc wireless networks using stochastic Petri nets.

In the literature, many works studied the performance of their proposed protocols for IoT. From the works that performed simulation for evaluating the performance, we find in the literature [151], [152], [153] and [154]. In [151], the authors proposed an enhanced version of Routing Protocol for Low-Power and Lossy Networks (RPL) for Internet of Multimedia Things. [152] proposed a new authentication protocol for Mobile IoT (MIoT) in order to construct a secure network. The protocol was tested and simulated as well as modeled using Alloy model. In [153], another performance analysis of IoT security protocols (IPSec [170] and Datagram Transport Layer Security (DTLS) [171]) was performed. The authors analyzed the impact of these protocols on the embedded devices resources. Furthermore, the authors in [154] proposed a new protocol and used OMNET++ simulator to test some properties such as end-to-end delay, energy efficient, etc, as well as they compared their proposal to traditional solution of RPL.

Many surveys were carried out in order to study and compare the protocols performance of IoT such as [172] and [173]. In [172], the authors made a survey by summarizing thoroughly the most relevant protocols and compared their work to other survey papers. In [173], another survey was made by performing an exhaustive analysis on existing mechanisms and protocols to secure communication on the IoT.

In term of studying formally IoT protocols, few works exist in the literature such as [155], [156], [174] and [157]. In [155], the authors developed two applications of test to study the performance of IoT web application. The first test application concerns the latencies of the communication protocols encodings and the performance of graphics renderings. The second test application measured the latency and the rate of message throughput of certain IoT messaging protocols for comparing their performances. The test has also been taken into consideration in [156] where the Internet of things application layer protocols Constrained Application Protocol (CoAP) [57] and MQTT were tested and their latencies were compared for high packet loss by creating a middle-ware component. In [174], the authors proposed a formal computational framework for publish/subscribe systems and studied the completeness and minimality properties. Talking also about publish/subscribe systems, the BonjourGrid[175] protocol which is based on these asynchronous systems was also studied in [157] by the formal modeling and verification using colored Petri nets.

The work presented in this chapter bridges these gaps by employing Probabilistic Timed Automata (PTA) and UPPAAL Statistical Model Checking (UPPAAL SMC) to formally model and verify:

- The Collision-Free Mobility Adaptive (CFMA) MAC protocol, specifically designed for mobile WSNs, addressing challenges of collision-free communication and adaptive energy efficiency.
- The MQTT 3.1.1 protocol, focusing on its robustness in dynamic IoT environments with mobility, ensuring critical properties like liveness, safety, and reliability, while analyzing performance metrics such as latency and throughput.

The contributions of this study are:

- Innovative Modeling Approach: This is the first formal study of the CFMA/MAC protocol, using PTAs to model the protocol's behavior under dynamic mobility scenarios. The robustness of the approach ensures that both static and mobile nodes are handled seamlessly.
- Comprehensive Verification Framework: Unlike existing works relying solely on simulation, this study uses UPPAAL SMC to quantitatively evaluate performance and formally verify properties, providing a stronger validation.
- Extension to IoT Protocols: This study extends formal methods to MQTT 3.1.1, demonstrating its adaptability and scalability in mobile IoT scenarios. The performance evaluation of crucial metrics ensures practical applicability.

The robustness of this work lies in its dual focus on formal verification and performance evaluation, combining rigorous mathematical modeling with real-world applicability. By addressing mobility, this study provides a robust framework for designing and analyzing future protocols in dynamic environments.

3.2 Formal Verification of Collision Free Mobility Adaptive Protocol for Wireless Sensor Networks

The main idea of CFMA/MAC protocol is the allocation of different backoff delays to the nodes.

The successful transmission of a frame begins with the coordinator assigning backoff delays based on node priorities. Nodes request association and send their priorities to the coordinator. Mobile nodes always have the highest priority and are allocated the shortest delays. If multiple nodes share the same priority, the first requester is assigned the shortest delay. *Wait_timers* are introduced to prevent collisions, particularly during the association phase when data might be transmitted. If a mobile node detects signals from another coordinator, it assumes it is moving towards an adjacent cluster and requests a delay from that cluster. Nodes with data to transmit must enter a backoff period, with delays assigned by the coordinator. After sending a Request to Send signal (RTS), the node waits for an acknowledgment. Following data transmission, the coordinator updates the delay values for nodes based on their priorities, ensuring fairness among all nodes. Therefore, this algorithm is divided into two phases: the association phase and the data transmission phase (running phase).

1. **Association phase:**

   In the association phase, nodes use the slotted CSMA/CA [176] protocol during the contention access period (CAP) to access the medium. Nodes send their priorities to the coordinator to receive different backoff delays. In CSMA/CA slotted, nodes follow these key steps [176]:

   - Initialization: Default parameters are set, including backoff period, contention window, maximum retries, and random backoff time.
   - Random Backoff: A random backoff time is chosen for the node to wait before attempting transmission.
   - Synchronization: Nodes align their backoff periods with the superframe to ensure enough time for transmission and acknowledgment.
   - Clear Channel Assessment (CCA): After the backoff time:
   - If the channel is busy, the node increments retry counters and may abandon the transmission if the retries exceed the maximum.
   - If the channel is clear, the node decrements the contention window. If the window reaches zero, the node sends its frame; otherwise, it retries after waiting for the next backoff period. This process ensures that nodes transmit only when the channel is clear, avoiding collisions.

2. **Running phase:**

   In this phase, the node with data (i.e., in its buffer) to be transmitted, must enter a backoff period before transmitting this data. Then, it sends Request to Send (RTS) signal and initializes the response timer to transmit data. Finally, An acknowledgement of the frame is sent from the coordinator with new values of backoff delays.

In order to facilitate the construction of the formal models, a semi formal modeling based on the Unified Modeling Language (UML)[177] is performed. This semi formal modeling is presented as sequence diagrams. These latter diagrams show clearly the interactions between the nodes and their coordinator. Figure 3.12 shows one of the sequence diagrams that we have made. This diagram depicts the main functionalities of the protocol in the running phase where it illustrates the necessary interactions for the transmission of a frame as described above.

3.2.1 Formal Modeling Using TA and PTA

This section presents the formal modeling of the CFMA/MAC protocol using Timed Automata (TA) and Probabilistic Timed Automata (PTA). The modeling concerns the behaviors of the five entities: the node (static or mobile) in the association phase, the static node in the running phase, the mobile node in the running phase, the coordinator and the medium. UPPAAL SMC allows to define the probability laws on the locality or the transition. In the models shown in Figure 3.4 and Figure 3.5, we associate a $rate = 1$ for any locality without invariant. In the probabilistic model of the medium, we used probabilities on transitions. We associate a probability of $1/10$ for each transition leading to a synchronous *fail* action. A probability of $9/10$ is

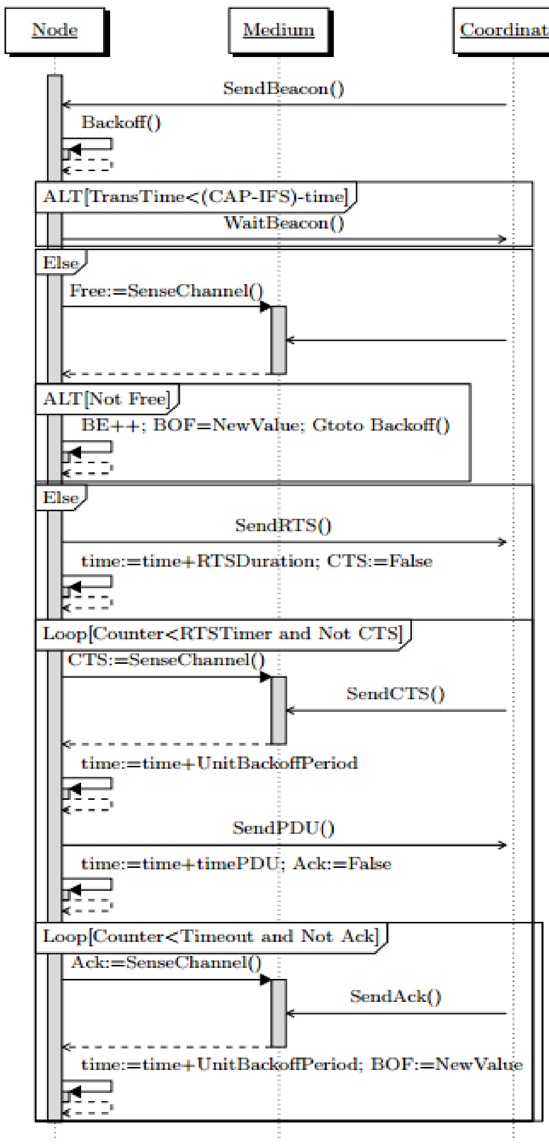

Node
Medium
Coordinator
SendBeacon()
Backoff()
ALT[TransTime<(CAP-IFS)-time]
WaitBeacon()
Else
Free:=SenseChannel()
ALT[Not Free]
BE++; BOF=NewValue; Gtoto Backoff()
Else
SendRTS()
time:=time+RTSDuration; CTS:=False
Loop[Counter<RTSTimer and Not CTS]
CTS:=SenseChannel()
SendCTS()
time:=time+UnitBackoffPeriod
SendPDU()
time:=time+timePDU; Ack:=False
Loop[Counter<Timeout and Not Ack]
Ack:=SenseChannel()
SendAck()
time:=time+UnitBackoffPeriod; BOF:=NewValue

Figure 3.1: Interaction during sending a frame -the running phase-.

associated for each transition leading to a transmission of a frame. These rates can be updated to show the response of the protocol for several values. In practice, such rates must be identified from statistics and experiments related to the nodes characteristics and the environment where these nodes are deployed.

3.2.1.1 Model of a Node in the Association Phase

Each node *nd* waits for *beacons* (control frames) for a period that does not exceed *Time_wait_beacon*, and if this is not the case the node receives a *fail*. The reception of a *beacon* is represented by a pair of synchronous actions (*BeginSendBeacon*, *EndSendBeacon*). The node, then, follows the slotted CSMA/CA algorithm to transmit a frame respecting the invariants of the localities. These invariants represent the transmission delay "*duration*", the delay to perform a CCA "*CCA_time*" and the delay to wait for an acknowledgement "*timeout*". A clock x, initialized to 0, is used on the incoming transitions of the localities containing these invariants. The node leaves the locality when its clock is equal to the threshold of the invariant or it receives a synchronous action. A synchronous action *Runing_Phase[nd]* represents the link between the association phase and the running phase of a node *nd*. Figure 3.2 illustrates the stochastic model of a node in the association phase (in UPPAAL SMC).

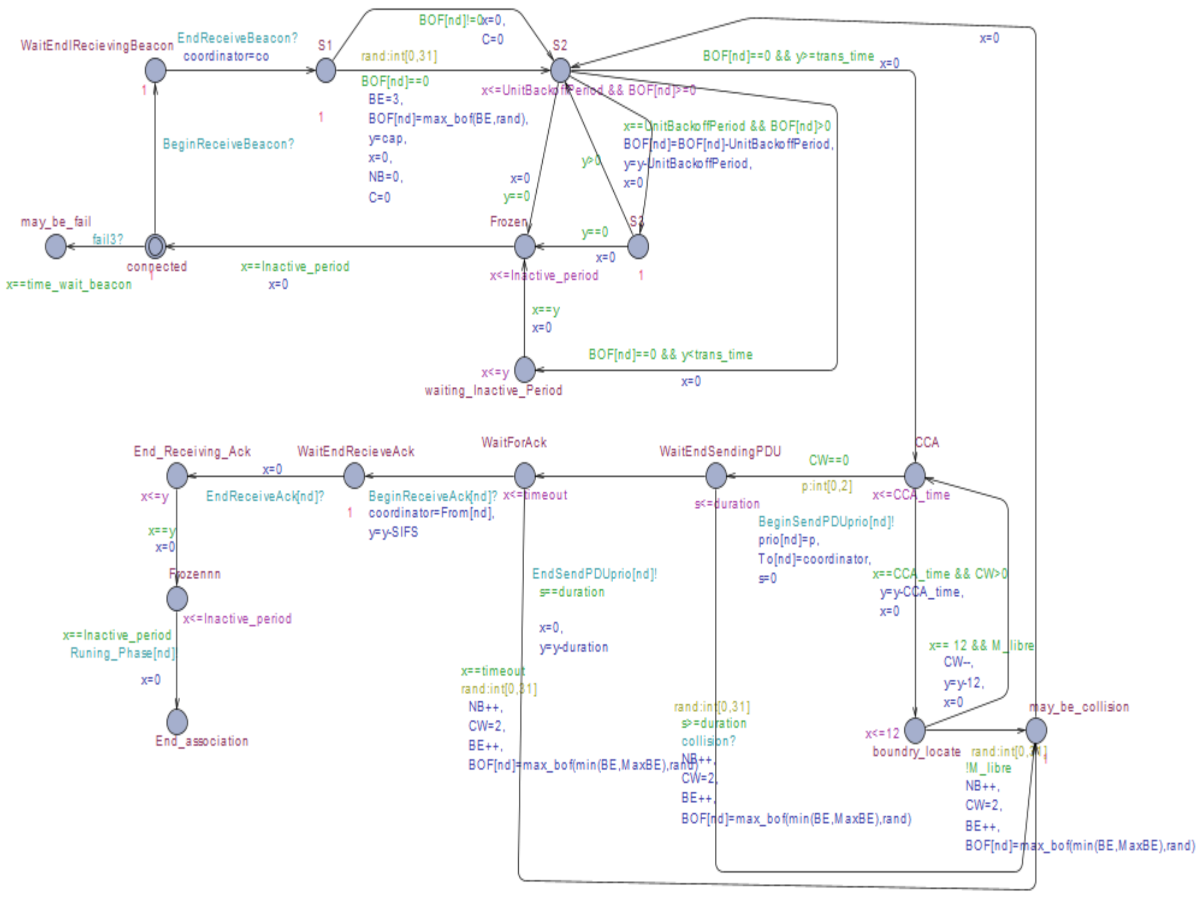


Figure 3.2: Node model in the association phase.

3.2.1.2 Model of a Static Node in the Running Phase

Once the node *nd* enters the execution phase by receiving the synchronous action *Running_Phase[nd]*, it waits for the *beacon*. It performs its *backoff* delay. Then, it tries to transmit its frame by sending respectively (*BeginSendRTS*, *EndSendRTS*) and (*BeginSendPDU*, *EndSendPDU*). After sending an *RTS* it receives a *CTS* which is represented by (*BeginReceiveCTS*, *EndReceiveCTS*) and after sending a *PDU* it receives an acknowledgement (*BeginReceiveAck*, *EndReceiveAck*) from the medium. Figure 3.3 illustrates the stochastic model of a static node in the running phase (in UPPAAL SMC).

3.2.1.3 Model of a Mobile Node in the Running Phase

A mobile node follows the same behavior as a static node except that it can migrate and change coordinators when it detects the signal strength of another coordinator. The mobility of a node is represented by the *Moving* locality. When it leaves this locality, it performs a change of coordinates after each reception of a *beacon*. The signal strength is modeled by the calculation, on the part of the medium, of the Euclidean distance between the node and the coordinators. Figure 3.4 illustrates the stochastic model of a mobile node in the running phase (in UPPAAL SMC).

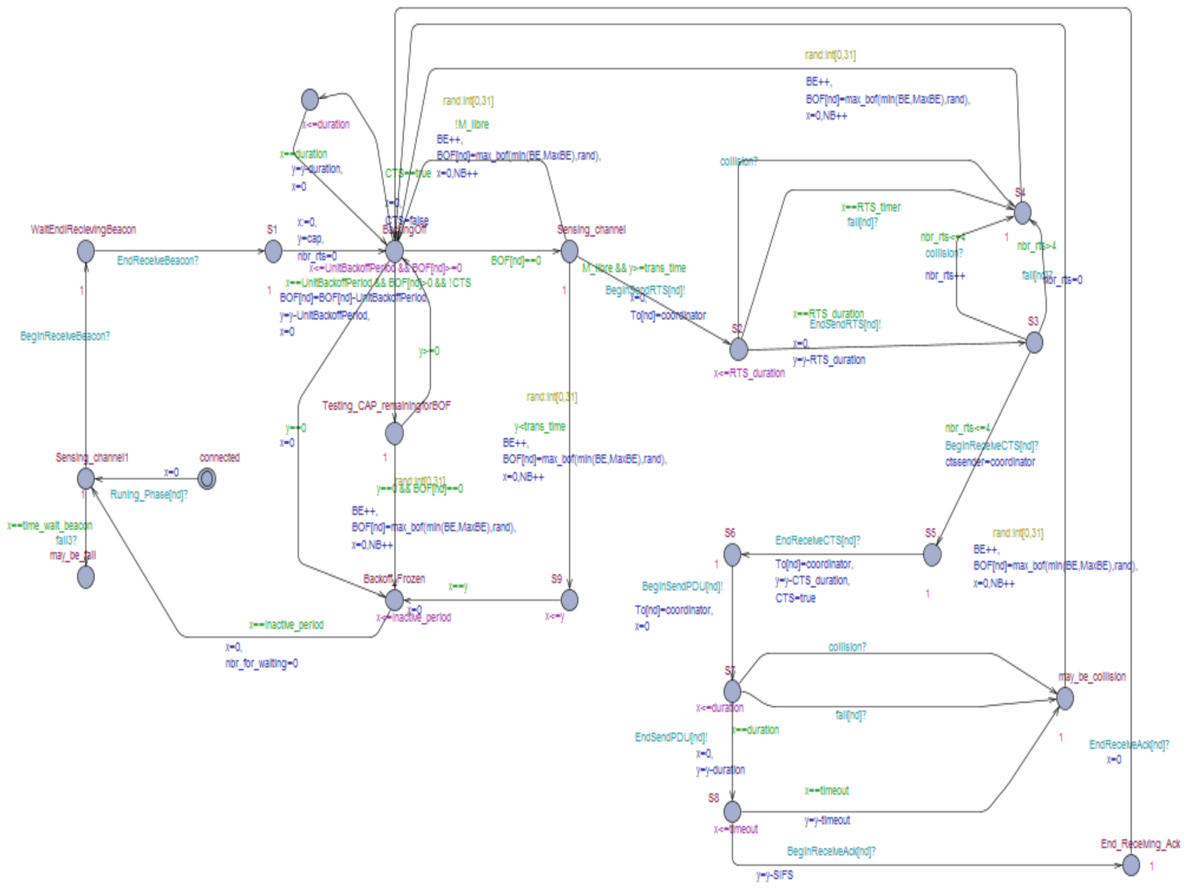


Figure 3.3: Static node model in the running phase.

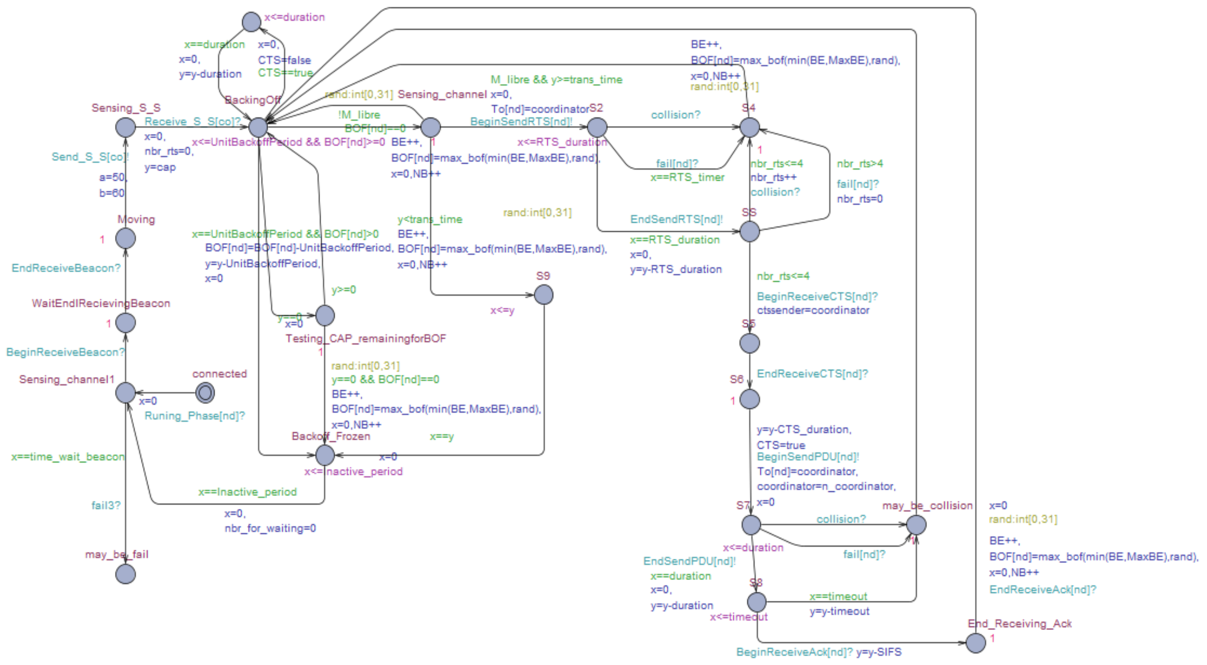


Figure 3.4: Mobile node model in the running phase.

3.2.1.4 Coordinator Model

The Contention Access Period *CAP* is the activity period of the coordinator. A coordinator *co* must listen to the medium during the *CAP*, if it receives an *RTS* or a *PDU*, it must send, respectively, (*BeginSendCTS*, *EndSendCTS*) or (*BeginSendAck*, *EndSendAck*). If the coordinator receives the identity (in an *RTS*) of the adjacent coordinator in the request sent by a mobile node then it sends *SendPrio[co]* to the adjacent coordinator in order to associate the requesting mobile node with this latter coordinater. Figure 3.5 shows the stochastic model of the coordinator (in UPPAAL SMC).

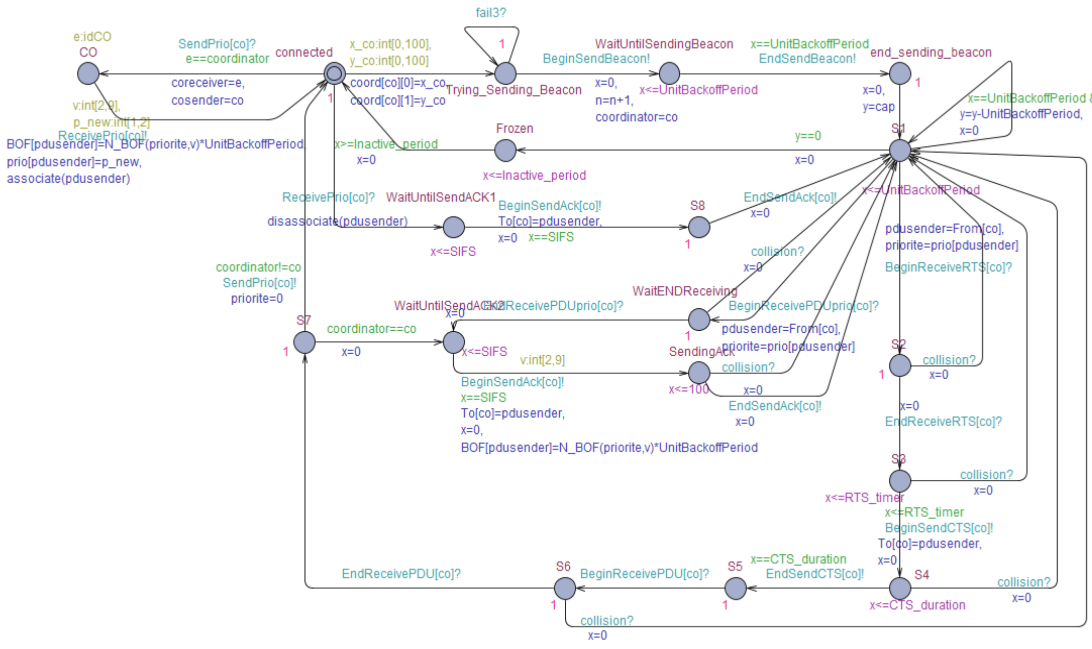


Figure 3.5: Coordinator model

3.2.1.5 Medium model

The medium model seems complicated (as seen in Figure 3.6) due to the number of synchronous actions between a coordinator and a node. After receiving a *BeginSend* synchronous action, the medium state will be occupied by changing a boolean *M_free* = *false* then it sends a *BeginReceive* or *fail* action. The same idea is done when it receives an *EndSend*, where it sends an *EndReceive* with *M_free* = *true*. When the medium receives *Send_S_S* with the coordinates of a mobile node, it calculates the distance between the coordinates of this node and the coordinates of the set of coordinators and then sends the identity of the new coordinator, which has the smallest distance, to the mobile node.

3.2.2 Qualitative Verification

We opt to verify some properties before assigning the probabilities to the localities and the transitions. Among the main classes of properties that can be expressed in temporal logics, we have studied the following properties using the query language provided in UPPAAL:

- Liveness property: a liveness property allows to express that a state or an event will necessarily occur in the future. we examine whether the node *nd_assoc0* will receive its acknowledgment.
- Safety property: this property is of the form: some bad thing will never happen. For example, whether a deadlock must never occur in the model.

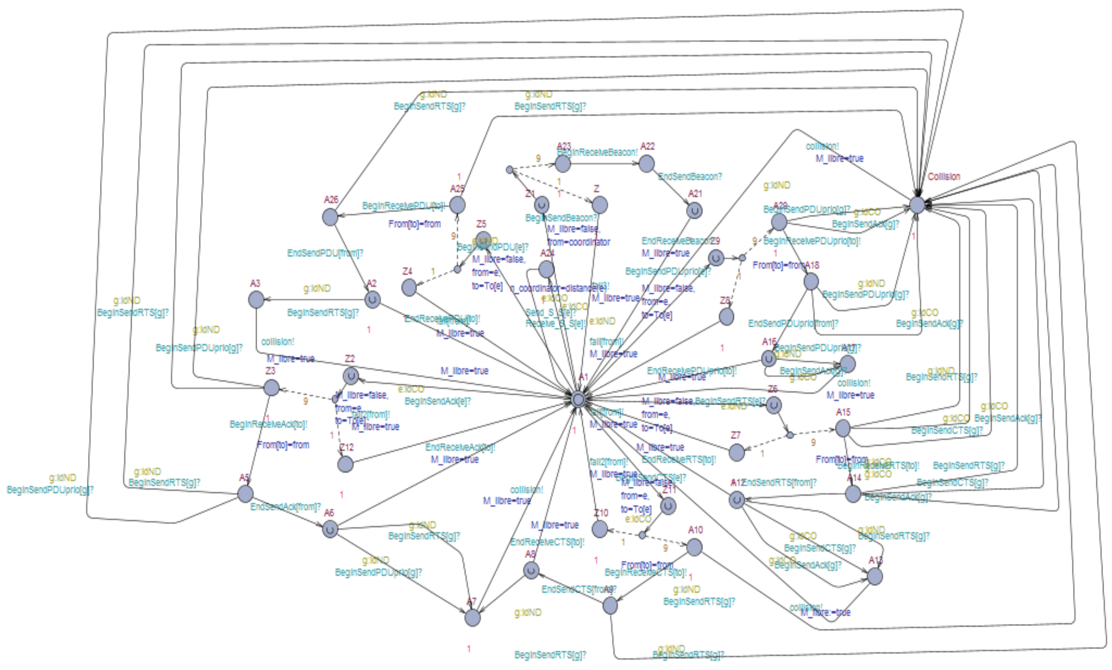


Figure 3.6: Probabilistic model of the medium.

Table 3.1: Satisfaction results of qualitative properties.

Property	***Query***	***Satisfied / Not satisfied***
Reachability 1	E<>$nd\_assoc0.End\_Receiving\_Ack$	Satisfied
Reachability 2	E<>$nd\_rp0.End\_Receiving\_Ack$	Satisfied
With T= 1000	E<>$(nd\_rp0.End\_Receving\_Ack\ and\ nd\_rp0.C <= 1000)$	Satisfied
With T= 80	E<>$(nd\_assoc0.may\_be\_collision\ and\ nd\_assoc0.C <= 80)$	Satisfied
Liveness	A<>$nd\_assoc0.End\_Receiving\_Ack$	Satisfied
Safety	A [] *not deadlock*	State space explosion

- Reachability property: we examine whether a node can finish the reception of its acknowledgment by the formula: $E<>$nd_assoc0.End_Receiving_Ack. We also exmine whether a mobile node can receive its acknowledgment.

In our case, we added a clock C to both node models (in association and running phases) to verify whether the nodes nd_assoc0 and nd_rp0 can reach the state "*end of reception of an acknowledgment*" before that the clock C reaches a threshold T $=$ 1000 and whether a node nd_assoc0 occurs a collision before a threshold T less or equal to 80. The verification results of the aforementioned properties are depicted in the Table 3.1.

Discussion: Due to the state space explosion in the classical model checking with UPPAAL, it becomes challeging to verify certain properties, such as the safety property shown in Table 3.1. The statistical model checker, the probabilistic timed automata and PCTL, all supported by UPPAAL SMC, solve this problem. After making the medium model probabilistic, we opt to perform a quantitative verification using SMC and PCTL for specifying the properties. This phase is presented in the following subsection.

3.2.3 Quantitative Verification

After the assignment of the probabilities to the localities and the transitions and making the models probabilistic. We can estimate the satisfaction rate (as seen in the Table 3.2) of each property by making the following queries: (i) $Pr[<= 100](<> nd\_assoc0.End\_Receiving\_Ack)$: this query examines whether a node can reach the state of $End\_Receiving\_Ack$. (ii) We also examine whether a node can finish its association (i.e., it can reach the state End_association). (iii) The query $Pr[<= 100](<> mnd\_rp0.End\_Receiving\_Ack)$ examines whether a mobile node can receive an acknowledgment.

Table 3.2: Results of some quantitative properties.

Query	***Probability interval***	***Confidence***
$Pr[<= 100](<> nd\_assoc0.End\_Receiving\_Ack)$	[0.811702, 0.911483]	0.95
$Pr[<= 100000](<> nd\_assoc0.End\_association)$	[0.902606,1]	0.95
$Pr[<= 100](<> mnd\_rp0.End\_Receiving\_Ack)$	[0.214981,0.314847]	0.95

UPPAAL SMC provides the possibility to draw the characteristic function of probability and hereafter some results with discussions.

We expect that the probability of finishing successfully the association decreases as the number of nodes increases. However, we note that this is not too clear due to the duration of the runtime. Figures 3.7 and 3.8 confirm the expected result by increasing the runtime value.

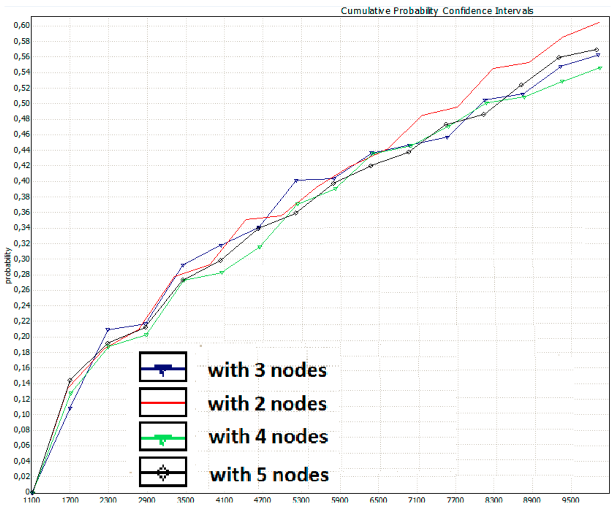


Figure 3.7: Probability of finishing the association depending on number of nodes with $runtime <= 10^4$.

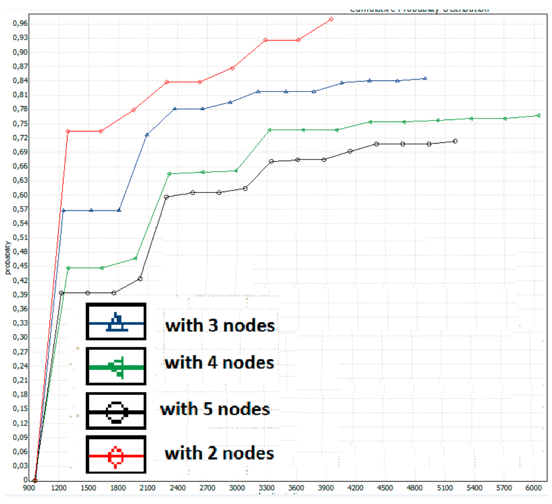


Figure 3.8: Probability of finishing the association depending on number of nodes with $runtime <= 10^5$

Figure 3.9 shows the probability of finishing a transmission by receiving an ACK, according to the number of nodes, for a duration of $runtime <= 10^3$. Likewise, we studied the probability of a collision depending on the number of static and mobile nodes in a runtime $<= 10^3$ as depicted in Figure 3.10.

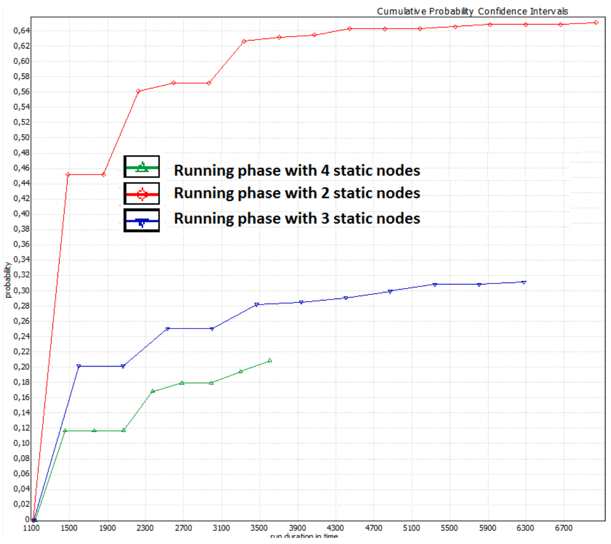


Figure 3.9: Probability of receiving Ack deppending on number of nodes.

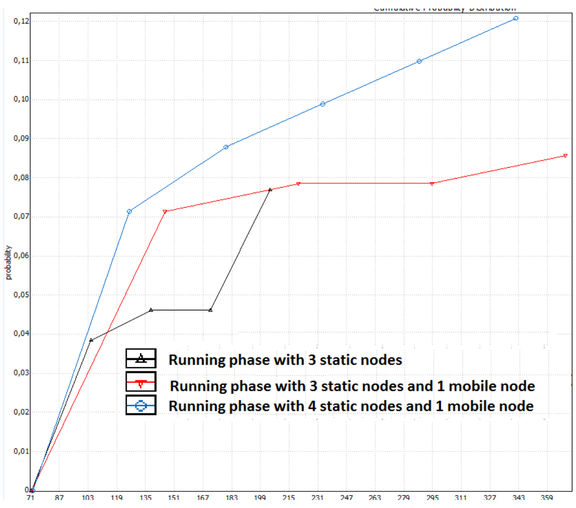
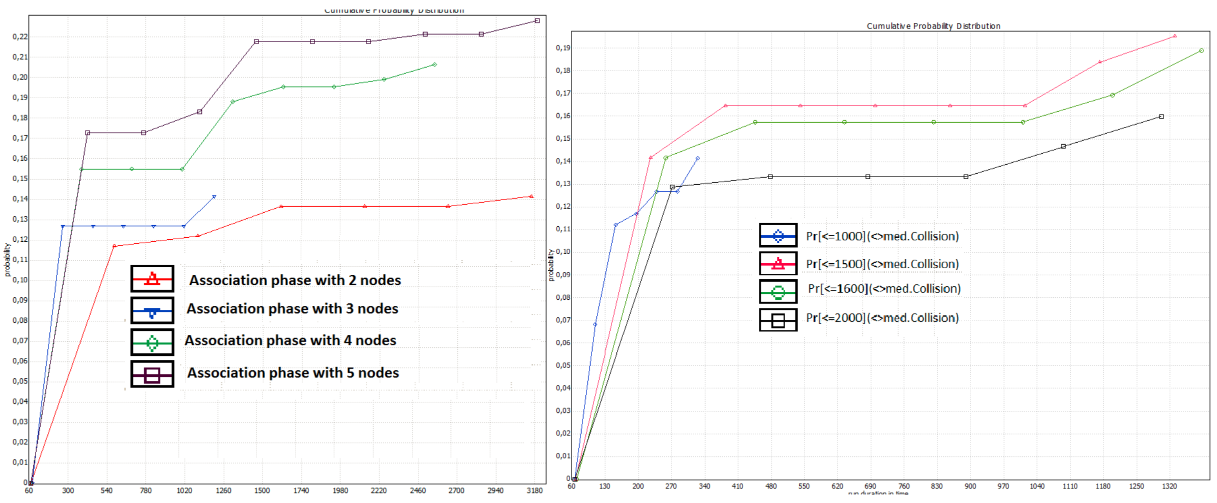


Figure 3.10: Probability of having collision depending on number of static and mobile nodes.

The probabilities of a collision depending on the number of nodes and depending on the runtime duration are depicted in Figures 3.11a and 3.11b, respectively.



(a) depending on number of nodes.

(b) depending on runtime.

Figure 3.11: Probability of having collision.

3.2.4 Discussion

The plots depicted in Figures 3.7 and 3.8 compare the probability of completing the association process under different runtime constraints ($runtime <= 10^4$ and $runtime <= 10^5$, respectively). In the first plot with $runtime <= 10^4$, The probability increases gradually with runtime, showing a consistent trend across different node counts. The curves are closely aligned, indicating similar behavior for different number of nodes. In the second plot with $runtime <= 10^5$, the probability of completion stabilizes at higher values, showing clearer distinctions between node counts. The association probability improves significantly for all cases, with fewer nodes reaching higher probabilities faster. Overall, increasing runtime enhances the probability of successful association, with fewer nodes consistently achieving better results.

The plot in Figure 3.9 illustrates the probability of receiving an acknowledgment increases over time, with different trends for different static nodes counts. In configuration with 2 nodes, the probability reaches a high value quickly, indicating efficient communication. In configuration with 3 nodes, it shows steady but slower increase, suggesting a more gradual acknowledgment process. In configuration with 4 nodes, it exhibits the slowest growth, likely due to increased complexity in message exchanges. Overall, fewer static nodes result in faster acknowledgment reception, while more nodes introduce delays but still trend toward higher probabilities over time.

The plot depicted in Figure 3.10 illustrates that the probability of collision increases over time for all configurations. In configuration with 3 static nodes, the probability shows a relatively steady increase. In configuration with 3 static nodes and one mobile node, it increases faster than the purely static setup, suggesting mobility introduces more contention. In configuration with 4 static nodes and one mobile node, it has the steepest increase, indicating that adding more nodes (both static and mobile) significantly raises the collision probability. Therefore, higher node density and mobility contribute to greater risk of collisions, affecting network performance.

The plots of Figures 3.11a and 3.11b present the probibility of collisions based on two factors: (i) Number of Nodes: As the number of nodes increases (from 2 to 5), the probability of collisions rises significantly and the sharpest increase occurs in the transition from 2 to 3 nodes, with diminishing gowth beyond 4 nodes. (ii) Runtime: Collision probability increases over time, with different probabilities for various time intervals and the probability stabilizes at longer durations but remains higher for lower thresholds (e.g., <=1000). Overall, a higher node count and longer runtime both lead to a greater likelihood of collision, reinforcing the impact of network density and prolonged activity on interference.

3.3 Formal specification, verification and evaluation of the MQTT protocol in the Internet of Things

Message Queue Telemetry Transport (MQTT) was invented by Andy Stanford-Clark and Arlen Nipper in 1999 and it is considered as an open-source Machine-to-Machine (M2M) protocol. MQTT 3.1.1 is a messaging protocol based on the publish/subscribe which qualifies it as an excellent candidate for communications within the Internet of Things.

MQTT 3.1.1 is a message publishing and subscribing protocol. Clients do not communicate directly with each other, they publish messages on a broker, and the messages are composed of a content and a topic. The broker stores the last message for each topic. Clients who are interested in a subject's messages can retrieve them by signing in to the broker. This solution has the advantage to allow for several clients to communicate even if they are never connected at the same time to the broker.

This communication is performed by exchanging a set of MQTT control packets. The following subsections present the format, the types of these packets and the levels of quality of services used by MQTT in its version 3.1.1.

3.3.1 Structure of MQTT 3.1.1 packets

This subsection describes the format of MQTT 3.1.1 control packets.

An MQTT 3.1.1 control packet consists of up to three parts, always in the following order: (i) Fixed header: all the control packets of MQTT 3.1.1 contains a fixed header, (ii) Variable header: some types of MQTT 3.1.1 control packets contain this header where the content varies according to the type of packet, (iii) Payload: it is considered as the last part of some MQTT 3.1.1 control packets.

3.3.1.1 MQTT 3.1.1 Control Packets Types

The MQTT 3.1.1 protocol works by exchanging a series of MQTT 3.1.1 control packets which are briefly recalled hereinafter.

- CONNECT is the first packet which must be sent to the server by the client after the establishment of a network by requesting a client a connection to a server. If a second CONNECT packet received by the server, the latter must processed it as a protocol violation and the client must be disconnected.
- CONNACK is a packet of acknowledgment connection request sent by the server in response to a CONNECT packet. It is the first packet sent from the server to the client. If the time of receiving CONNACK packet exceeds a reasonable amount of time, the client should close the network connection.
- PUBLISH is the packet which can be sent by both server and client to transport an application message.
- PUBACK is a packet of PUBLISH acknowledgment with the quality of service (QoS) level.
- PUBREC is the response to a PUBLISH packet with QoS 2.
- PUBREL is the response to a PUBREC packet.
- PUBCOMP is the response to a PUBREL packet.
- SUBSCRIBE can be sent by a client to create one or more Subscriptions which registers a client's interest in one or more topics. The SUBSCRIBE packet also specifies (for each Subscription) the maximum QoS with which the server can send application messages to the client.
- SUBACK is a packet of SUBSCRIBE acknowledgment that confirms the reception and processing of a SUBSCRIBE packet.
- UNSUBSCRIBE can be sent by a client to unsubscribe from topics.
- UNSUBACK is a packet of UNSUBSCRIBE acknowledgment that confirms the reception of an UNSUBSCRIBE packet.
- PINGREQ can be sent by a client when there is no control packet to be sent to the server. This packet indicates to the server that the client is alive. It is also used for exercising the network to indicate that the network connection is active.

- PINGRESP can be sent by the server in response to a PINGREQ packet. It indicates that the server is alive.
- DISCONNECT can be sent by a client as the last control packet to the server in order to indicate the clean disconnection of the client.

3.3.1.2 Quality of Service Levels

The delivery protocol of MQTT 3.1.1 considers both server and client as sender and receiver. The delivery of an application message can be either from a single sender to a single receiver or from a server to more than one client. MQTT 3.1.1 delivers application messages according to the Quality of Service (QoS) levels defined hereinafter.

- QoS 0 (At most once delivery): the sender must send a PUBLISH packet with QoS=0, DUP=0 and the receiver accepts ownership of the message when it receives the PUBLISH packet.
- QoS 1 (At least once delivery): the sender must assign an unused packet identifier, must send a PUBLISH packet with QoS=1, DUP=0, and must treat the PUBLISH packet as "acknowledged" until it has received the corresponding PUBACK packet from the receiver.

  On the other side, the receiver must respond with a PUBACK packet and must treat any incoming PUBLISH packet that contains the same Packet Identifier.
- QoS 2: the sender must do the same delivery protocol of QoS 1 with the difference in sending the PUBLISH packet with QoS=2, DUP=0. In addition, it must send and treat the packet PUBREL when it receives PUBREC. The receiver must respond with a PUBREC packet. When receiving PUBREL, it must not cause duplicate messages to be delivered and it must respond to a PUBREL packet by sending a PUBCOMP packet containing the same Packet Identifier as the PUBREL.

3.3.2 Informal Description of the MQTT 3.1.1

MQTT 3.1.1 follows an operating mode which is explained in the following subsections.

3.3.2.1 Connection/Disconnection

MQTT 3.1.1 uses persistent connections between clients and the broker, and it uses network protocols that guarantee a high level of reliability such as TCP. Before sending control commands, a client must first register with the broker, which is done with the CONNECT command. Various connection parameters can then be exchanged such as the client identifier or the desired persistence mode. The broker must confirm to the client that the registration has been taken into account, (i.e., it indicates that an error has occurred by returning a CONNACK accompanied by a return code). There is a PINGREQ command to let the broker know that the client is still active, the broker will respond with a PINGRESP to tell the client that the connection is still active. When the client wants to disconnect, it sends a DISCONNECT command to the broker. Otherwise, the broker will consider the disconnection to be abnormal and will send accordingly the message of *WILL* on behalf of the client to all the subscribers.

3.3.2.2 Subscriptions and Publications

Each published message is necessarily associated with a topic, which allows its distribution to the subscribers. Topics can be organized in tree hierarchy, so subscriptions can be based on filtering patterns.Subscription management is very simple and consists of three essential commands:

- SUBSCRIBE: Allows a subscriber to subscribe to a topic, once subscribed,it will then receive all the publications concerning this topic. A subscription, also, defines a quality of service level.The successful reception of this command is confirmed when the broker sends a SUBACK carrying the same packet identifier.
- UNSUBSCRIBE: Gives the possibility to cancel a subscription, and thus, no longer receives subsequent publications. The successful reception of this command is confirmed when the broker sends a UNSUBACK carrying the same packet identifier.
- PUBLISH: Initiated by a client, it allows to publish a message which will be transmitted by the broker to the possible subscribers. The same command will be sent by the broker to the subscribers to deliver the message.

If the required quality of service is greater than zero, messages will be exchanged to confirm the publication support.

3.3.3 Formal Modeling and Verification of MQTT 3.1.1

This section presents the semi-formal modeling based on UML[177], the formal modeling and the formal verification using UPPAAL SMC.

3.3.3.1 Semi Formal Modeling

To understand the operation of the MQTT 3.1.1 protocol and as a first step we opt for semi formal modeling of the protocol, which is performed using UML language diagram. This diagram is the sequence diagram to show the interactions between the clients and the broker.

Figure 3.12 shows the protocol sequence diagram. This figure shows a summary of the necessary interactions for the transmission of a frame according to the protocol already described. These interactions involve three entities: the publisher, the subscriber and a broker. At this stage, the phase of semi formal modeling of the main principles of the protocol is established. In the following subsections, we use UPPAAL SMC [178] for the formal modeling and verification of the protocol MQTT 3.1.1.

3.3.3.2 Formal Modeling Using Stochastic Timed Automata

The stochastic models of the entities that participate in the communication using this protocol are shown in Figures 3.13a, 3.13b, 3.14a, 3.14b, 3.14c, 3.15 and 3.16.

3.3.4 Formal Verification of MQTT 3.1.1

We opt to verify some qualitative properties, such as reachability, safety and liveness, using the query language provided in the UPPAAL.

Verifying whether a subscriber and a publisher can arrive at an initialization state (S18 and P9 respectively) is formulated by the formulas: $E{<>}sub1.S18$ and $E{<>}pub1.P9$, respectively.

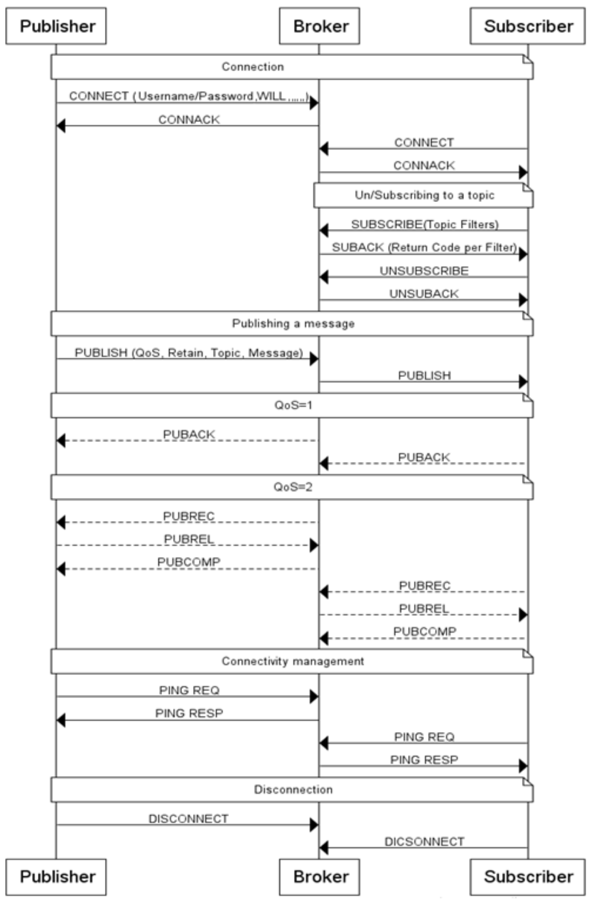
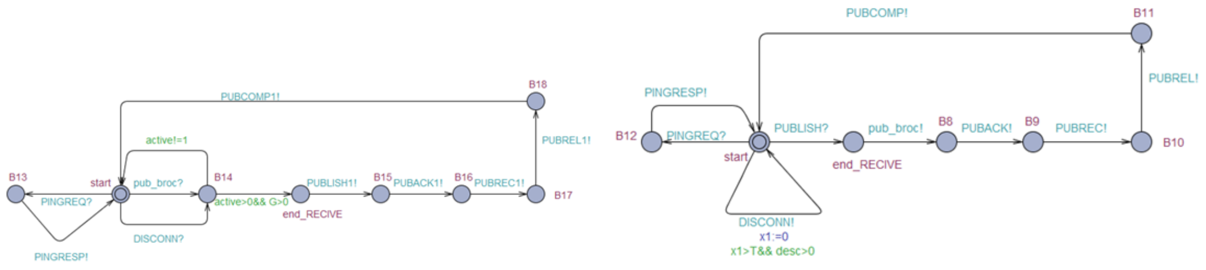


Figure 3.12: Sequence diagram of MQTT 3.1.1



(a) Side of Subscriber. (b) Side of Publisher.

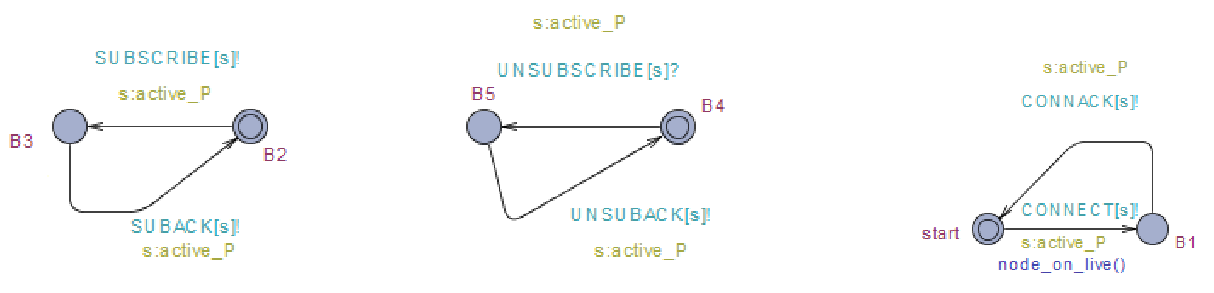
Figure 3.13: Model of the broker

Likewise, the liveness property is verified using the following query: $A<>publish\_sub$.END_RECEIVE. In addition, the safety property is verified using the formula: A [] $not\ deadlock$ which means that a deadlock must never occur in the model.

UPPAAL SMC gives us the possibility of adding temporal constraints to the formulas of the properties to be verified. Using this query language, we have examined whether the subscriber can receive a publication before that the waiting time exceeds or the value of X reaches a certain threshold where X is a clock added to the model of the broker in the publication phase at the publisher side, in order to present the "keep alive" property. We opt to verify this later property with the thresholds T = 45000, 1000 and 100.

The results obtained for the aforementioned properties are depicted in Table 3.3.

Discussion: In some cases, it becomes challeging to verify certain properties such as safety



(a) (b) (c)

Figure 3.14: Models of (a) Subscription, (b) Unsubscription and (c) Connexion.

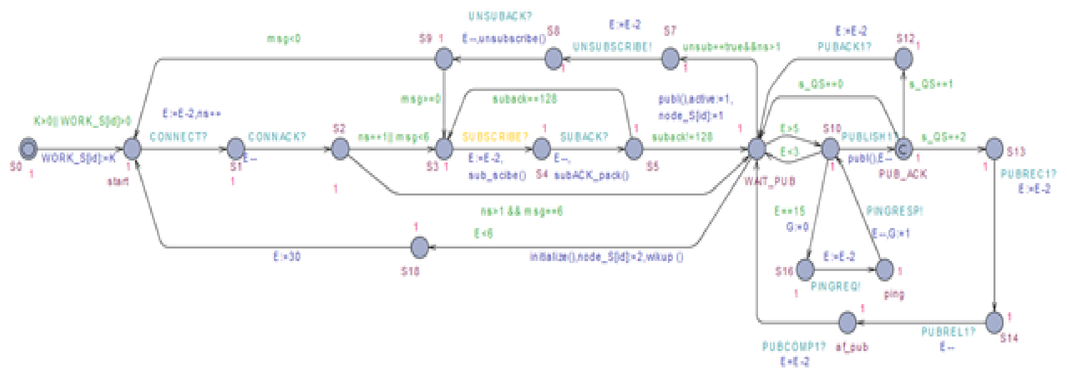


Figure 3.15: Model of the subscriber.

(as seen in Table3.3) and the reachability, in the case of 3 subscribers and 3 publishers. And that is due to the state space explosion in the model-checker. However the utilization of UPPAAL SMC and probabilistic CTL in specifying the properties, solve this problem. Thus the property examined before (with 3 subscribers and 3 publishers) becomes satisfied with probability interval as seen in Table 3.4. This later table demonstrates also the subscriber probability of arriving at an initialization phase with a runtime lower or equal to 100 in the case of 9 publishers and 9 subscribers.

We opt to draw (as seen in Figure 3.17) the cumultative probability confidence interval of finishing the reception of a publication according to the number of nodes for a runtime $<= 10^3$ units of time. Likewise, the probability that a publisher disconnects and that its subscribers receives the *will* during a runtime $<= 10^3$ time units also studied and the results are depicted in

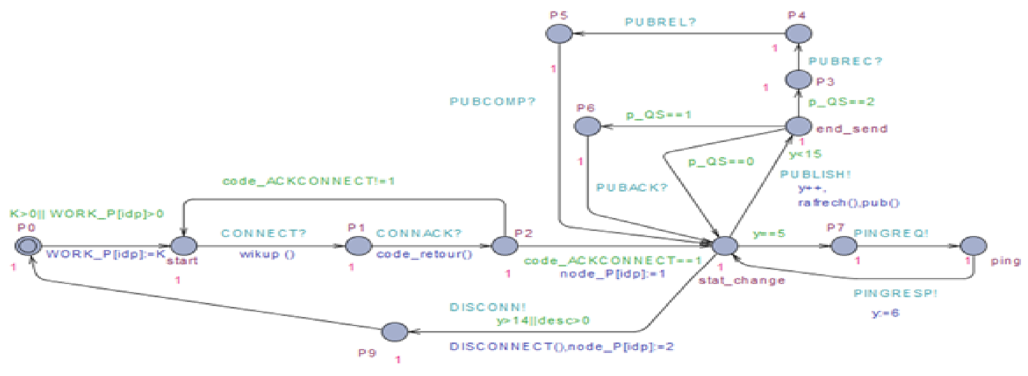


Figure 3.16: Model of the publisher.

Table 3.3: Satisfaction results of qualitative properties.

Property	***Query***	***Satisfied / Not satisfied***
Reachability 1	$E{<>}sub1.S18$	Satisfied
Reachability 2	$E{<>}pub1.P9$	Satisfied
With T= 45000	$E{<>}(sub1.PUB\_ACK\ and\ pub\_publish.X1 < 45000)$	Satisfied
With T= 1000	$E{<>}(sub1.PUB\_ACK\ and\ pub - publish.X1 < 1000)$	Satisfied
With T= 100	$E{<>}(sub1.PUB\_ACK\ and\ pub\_publish.X1 < 100)$	Satisfied
Liveness	$A{<>}publish\_sub$	Satisfied
Safety	$A\ [ ]\ not\ deadlock$	State space explosion

Table 3.4: Results of some quantitative properties.

Query	***Satisfied / Not satisfied***	***Probability interval***	***Confidence***
Pr[<=100]<>sub1.S18	Satisfied	[0.303567,0.403432]	0.95
Pr[<=100]<>(NNI_P>1 and NNL_P>=2)	Satisfied	[0.902606,1]	0.95

Figure 3.18.

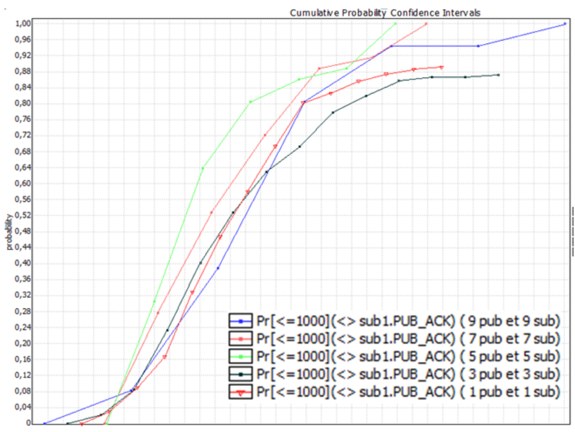


Figure 3.17: Cumultative probability of a successful publication depending on the number of nodes.

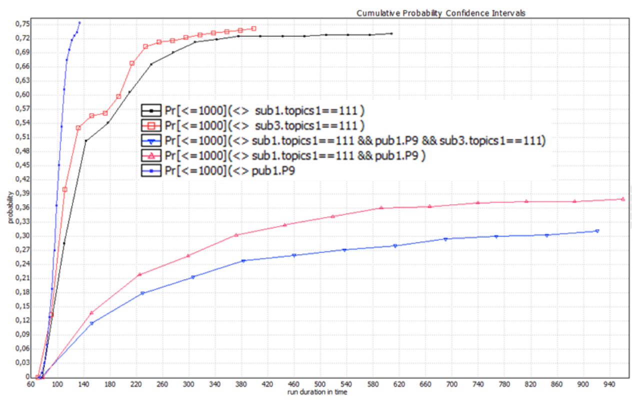
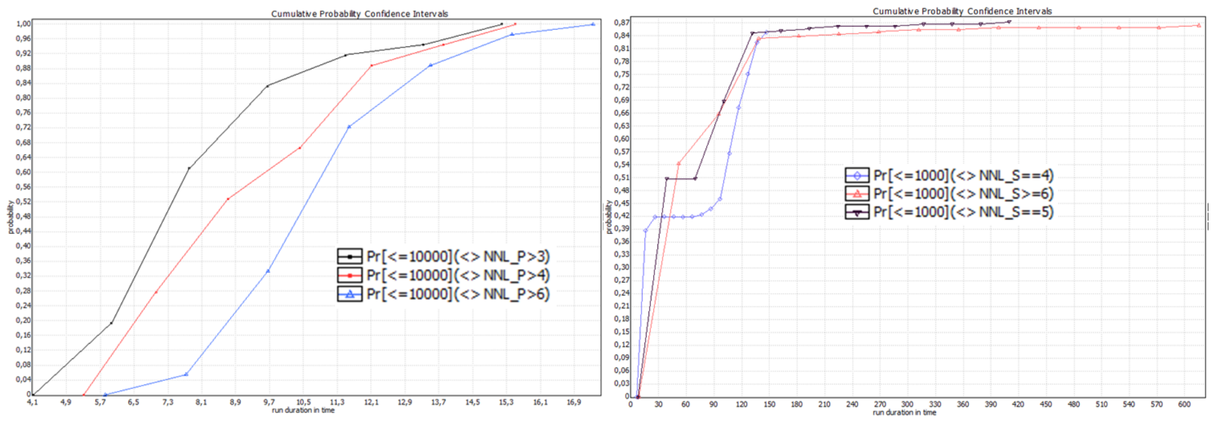


Figure 3.18: Cumulative probability of publisher disconnection with the successful reception of the *will* from subscribers.

Figure 3.19a and Figure 3.19b show the probabilities of having active publishers and active subscribers for a runtime $<= 10^4$ and 10^3 units of time, respectively.



(a) For a runtime $<= 10^4$ time units (b) For a runtime $<= 10^3$ time units

Figure 3.19: Probability of having active publishers and subscribers.

3.3.5 Discussion

The plot in Figure 3.17 illustrates the cumulative probability of a successful publication acknowledgment (PUB_ACK) as a function of the number of nodes involved in the process. The different curves represent varying numbers of publishers and subscribers, demonstrating how the publication success rate evolves with network density. From the observed trends, the probability of a successful publication increases as the number of nodes grows. Configurations with a higher number of publishers and subscribers tend to exhibit a faster convergence towards higher success probabilities. This indicates that a denser network enhances the likelihood of successful PUB_ACK due to increased redundancy and multiple paths for message transmission. However, variations among the curves suggest that beyond a certain threshold, adding more nodes does not necessarily lead to a proportional improvement in publication success. This could be due to network congestion, increased collisions, or limitations in the underlying protocol. The confidence intervals highlight the range of uncertainty in the measurements, reinforcing the reliability of the observed trends. Overall, these results emphasize the importance of balancing network density and communication overhead to optimize publication success rates in IoT-based communication systems.

The figure 3.18 illustrates the cumulative probability of publisher disconnection while ensuring that subscribers successfully receive the "will" message. In MQTT, the "*will*" message is a Last Will and Testament (LWT) feature, ensuring that if a publisher disconnects unexpectedly, a predefined message is sent to subscribers. The different probability curves indicate how various conditions (e.g., topic subscriptions and publisher states) affect the likelihood of successful message reception upon disconnection. The results provide insights into MQTT reliability, particularly in handling unexpected failures.

Figures 3.19a and 3.19b present cumulative probability distributions for having active publishers and subscribers within specified runtime limits: (i) The plot of Figure 3.19a shows the probability of having more than a certain number (NNL_P > X) of active publishers over time. The different curves correspond to thresholds (3, 4, and 6). The probability increases as time progresses, stabilizing near 1. (ii) The plot in Figure 3.19b illustrates the probability of maintaining a certain number (NNL_S = X) of active subscribers. The different lines indicate thresholds (4, 5, and 6). The probability grows rapidly and reaches saturation. These results highlight system stability, showing how long it takes for a reliable number of publishers and subscribers to be active.

3.4 Conclusion

In conclusion, this chapter has presented a detailed examination of the application of formal modeling and verification techniques to reconfigurable communication systems. The two main contributions (evaluating the CFMA/MAC protocol for mobile wireless sensor networks and analyzing the MQTT 3.1.1 protocol for IoT systems) demonstrate the power of formal methods in ensuring both the reliability and performance of protocols operating within reconfigurable environments. By incorporating probabilistic timed automata and leveraging the UPPAAL SMC tool, we have been able to verify not only the correctness of the systems but also their performance under various operational scenarios.

Looking ahead to the next chapter, we will shift focus from communication protocols to another domain of reconfigurable systems: reconfigurable manufacturing systems (RMS). In this domain, dynamic reconfigurability and the need to optimize performance under ever-changing operational requirements present unique challenges. The upcoming chapter will introduce a novel approach combining Petri nets, their extension as Reconfigurable Object Nets, and multi-objective genetic algorithms to address these challenges. This approach aims to provide a formal framework that accounts for dynamic reconfigurability while optimizing the system's performance which is an important step forward in understanding and improving reconfigurable systems in manufacturing settings.

Chapter 4

On Formal Modeling, Analysis and Optimization of Reconfigurable Manufacturing Systems (RMSs)

4.1 Introduction

Currently, companies are vigorously pursuing strategies to maintain competitiveness by providing rapid and cost-effective responses to dynamic and fluctuating markets. To achieve this goal, there is a growing trend towards the adoption of Reconfigurable manufacturing systems (RMSs). RMSs represent the cutting-edge in manufacturing technology enabling rapid adjustments in both hardware and software system configurations to meet dynamic customer demands of costumers and effectively handle unpredictable failures for the system protection [179].

The hardware reconfiguration involves physical activities such as adding/removing machines or machine modules, altering layouts and changing material handling devices [180]. However, these activities can result in costly and time-consuming reconfigurations [181]. On the other hand, the software reconfiguration, also known as logical reconfiguration, consists of adjusting planning, route, schedule [180]. Such activities are widely adopted to increase operational capacity [181].

This chapter focuses on the application of formal modeling and analysis techniques, as well as optimization methods, in designing reconfigurable manufacturing systems (RMSs). It also emphasizes ensuring that optimal configurations are verified during the reconfiguration process.

The rest of this chapter is organized as follows: Section 4.2 presents the related works and context of the work. Section 4.3 introduces the formal background of the proposed formalism. Section 4.4 gives the informal and formal definitions of Gen-RONs and how it is equipped with optimization abilities inspired from genetic algorithms. Section 4.5 introduces the proposed algorithms used in Gen-RONs and analyzes their complexities. Section 4.6 shows how Gen-RONs are used to specify and optimize an RMS case study, and finally Section 4.7 concludes this chapter and discusses the future work.

4.2 Context of the Work

An RMS is considered as a set of reconfigurations, each defined by its unique structure and properties. The reconfiguration process in an RMS involves evolving from one configuration to another. Consequently, it is crucial to (i) preserve the good properties of the previous configuration (such as reliability, performance, deadlock-freeness, etc.), (ii) achieve an optimal configuration (in terms of optimal cost, system adjustment, etc.) and (iii) ensure reliable scheduling. These factors must be rigorously checked and satisfied after each reconfiguration step throughout the lifespan of an RMS.

Recent advancements in RMS technology have sparked a surge of innovative research endeavors such as [182], [183], [184], [185], [186], [187]. The work reported in [182] identifies key RMS enablers (RMSEs) through an extensive literature review and expert input. The study develops a structural framework using hybrid methods (Robust Best Worst method and Interpretive Structural Modeling) to analyze and categorize enabler interactions to guide RMS implementation and facilitate its adoption in industry. RMSEs' study is also discussed in [183]. It reviews literature on reconfigurability enablers in manufacturing by consolidating fragmented concepts, providing classification frameworks for system components and identifying new industry 4.0 enablers. The research outlined in [184] proposes decentralized decision-making strategies to enhance RMSs' adaptability, responsiveness and sustainability. The work reported in [185] highlights the importance of RMS in aerospace. It introduces an Ontology-based engineering (OBE) methodology called Models for Manufacturing (MfM). This method integrates design processes and preserves company knowledge. The study in [186] proposes a systematic methodology by combining existing and new development tools for the design and development of changeable and reconfigurable manufacturing systems to enhance competitiveness in a dynamic industrial environments. The Digital Twin technology is integrated with RMSs in [187] to address new challenges in manufacturing. The study presents a Digital Twin Monitoring and Simulation Integrated Platform (DTMSIP) that enables real-time online supervision and high-fidelity offline simulation of RMS production. These studies have primarily focused on operational adaptability and system integration within specific industrial contexts.

While existing researches have extensively explored various dimensions of RMSs, there remains a significant opportunity to integrate advanced formalisms and optimization techniques systematically.

Petri nets are a formalism used in the modeling, verification and diagnosis of concurrent and distributed systems [188], [189], [190], [191], [192]. Using formal methods in RMSs design has attracted many researchers in the field. In literature, numerous studies have utilized Petri nets [193] and their extensions to analyze configuration properties or formalize the reconfiguration process, including works such as [194], [179], [195], [196], [197], [67], [198], [199], [200], [201], [202], [203], [204], [205], [206], [207], [208], [209], [210], [211], [212], [213], [214], and [215]. On the other hand, to study the performance of reconfiguration process, several studies have employed evolutionary techniques such as genetic algorithms [10] to model the evolution process in RMSs and optimize their performance, including works such as [216], [217], [218], [219], [220], [221], [222], [223],[224], [225], [226], [227], [228].

The aforementioned research studies have focused on either the optimization problems such as using evolutionary algorithms or the formalization problems using Petri nets. Existing literature includes studies that have effectively addressed formal specification and optimization techniques in manufacturing using a unified formal method such as [229], [230], [231], [232], [233], [234], [235], [236] and [237]. The formal method used in [229], [230], [231] and [232] is first-order hybrid Petri nets (FOHPNs) [238]. This formalism is an extension of Petri nets that designs, analyzes and optimizes manufacturing systems using time-driven and event-driven dynamics and traditionally linear programming approach. In [233], simulated annealing is combined with stochastic PNs for the modeling, evaluation and optimization in manufacturing systems. The work reported in [234] proposes an approach based on Petri nets and A* search algorithm for modeling and optimizing manufacturing systems. It uses Petri nets to model blocking constraints, job routings and resources and A* for scheduling. The work reported in [235] tackles scheduling optimization problem in flexible job shop using PNs and local search algorithms. The work outlined in [236] proposes a combination of mixed integer programming (MIP) and a branch-and bound algorithm (B&B) using timed Petri nets to model FMSs and optimize the scheduling. The approach includes methods to minimize the makespan of FMSs. The work reported in [237] addresses the optimization of dynamic scheduling in unreliable flexible manufacturing systems (UFMSs) through predictive maintenance using Internet of Things (IoT) data. It utilizes timed Petri nets to model the systems and formulate mixed-integer linear programming (MILP) instances. This study aims to prevent equipment failures, minimize downtime and improve reliability. The combination of evolutionary algorithms particularly genetic algorithms (GAs) with Petri nets in solving manufacturing systems optimization and formal specification problems has garnered significant research interest such as works in [239], [240], [241], [242], [243], [244], [245], [246], [247], [248], [249], [250], [251], [252], [253] and [254].

The work reported in [239] presents a hybrid approach that combines queueing Petri net (Q-PN) and the genetic algorithm. It has been proposed for the formal modeling and performance evaluation of manufacturing systems and for producing an optimal scheduling policy.

The study in [240] models the resource allocation by colored Petri nets (CPNs) to obtain the near-optimal one and the event-driven schedule, and the operation schedule is done through a genetic algorithm. In [241], the scheduling problem was solved using CPNs and GAs in FMSs. The research outlined in [242] resolves the scheduling problem of semiconductor manufacturing systems using colored timed Petri nets to model generalized schedule generator and GAs to develop a scheduling strategy provider. In [244], flow-line system Petri nets (FPNs) are combined with GAs to find optimal schedule to avoid deadlocks in automated guided vehicle jobs. Evolutionary PNs (EPNs) are introduced where places can be added or deleted to define mutation and crossover for PNs in [246]. The work reported in [246] applies EPNs in reverse engineering of biochemical reaction networks instead of RMSs. The combination of PNs and GA in FMSs scheduling is reported in [243, 249]. In [249], simplified timed Petri nets (Simplified TPN) is introduced and new mutation and crossover operators are defined. PNGA (Petri net and genetic algorithm) and Improved PNGA are introduced, respectively, in [245] and [247]. The two formalisms are used to estimate, simulate and optimize flexible job shop. Process planning nets (PP-nets) have been introduced and combined with GAs in [248] for scheduling manufacturing systems. The work reported in [250] is oriented towards business processes and uses PNs and GAs to optimize resource allocation. However, the reconfiguration is not considered. The work [251] explores supervisor simplification in automated manufacturing using GAs within discrete event systems. It develops optimal simplification methods using GA for supervisors under fixed and variable specification parameters, emphasizing Petri nets for modeling. The research work presented in [252] focuses on optimizing flexible job shop schedules by integrating PNs and GAs. This work employs a single-objective GA to minimize makespan. It deals with static scheduling problems rather than dynamic reconfigurations. Flexible job shop scheduling challenges is also addressed in [253]. The study models production systems using timed coloured Petri nets to describe flexible computerized numerical control (CNC) machines. It uses a Particle Swarm Optimization (PSO) algorithm to optimize the job type sequencing and throughput in the manufacturing system. While this study utilizes PSO as a well-known meta-heuristic approach, it does not employ genetic algorithms. Although both PSO and GAs algorithms fall under the category of meta-heuristics and used for optimization in complex problems, they are distinct techniques with different mechanisms. The work outlined in [254] addresses the challenges of automated guided vehicles (AVGs) scheduling and processing sequence conflicts (PSC) in no-buffer assembly lines. The study proposes a Petri net model to handle these issues and introduces a genetic algorithm-based look-ahead scheduling algorithm to optimize the makespan, enhancing flexibility and efficiency assembly lines. This study remains focused on FMSs, considering that assembly lines are crucial type of FMSs.

The above research considered the combination of optimization techniques (particularly GAs) and PNs; however, the following issues arise: (i) all the above works focus on FMSs and classical manufacturing systems rather than RMSs, (ii) the scheduling algorithms proposed in the aforementioned works are limited to the classical schedule problems and are therefore not suitable for RMSs, (iii) the use of Petri nets is primarily for simulation purposes, (iv) Some works are characterized by a specific methodologies, limiting their application to other frameworks' problems, (v) certain research studies focus on a single objective in their optimization approaches, such as makespan, cost, due-date, number of machines, tardiness, etc.

Few works have explored the formal study and optimization of RMSs such as [255], [256] and [257]. The work reported in [255] proposes a genetic algorithm combined with deterministic timed-place Petri nets for reconfigurable production lines (RPLs). It utilizes PNs to model the behaviors of RPLs. The crossover and mutation operators are oriented towards the elements of Petri nets. The authors use weighted-sum method to study the optimization criteria. However, this approach has drawbacks, such as its inability to detect optimal solutions in non-convex regions. Additionally, the absence of formal definition for mutation and crossover operators within the Petri net framework. The process of restructuring (or the rearrangement) required to define these operators within PNs remains undefined, thus lacking a solid mathematical foundation. This lack of formalization hinders the decisions regarding which properties (both quantitative and qualitative) should be preserved from one generation to the next generation during the evolution process.

In [256], a distributed approach based on multi-agent system (MAS) is presented for studying the capacity scalability problem (CSP) in RMS. PNs are employed to model the processes and interactions of agents within the RMS. The CSP is formulated as an optimization problem, and classical Lagrange relaxation optimization theory is applied to solve it. The contract net protocol (CNP) is utilized for task allocation and coordination among agents, thereby minimizing the number of required resources and ensuring timely completion of operations. The specific optimization methods used in solving operation agent scheduling problem (OASP) are not thoroughly detailed which limits the understanding of the efficiency and robustness of the optimization process. In addition, the distributed nature and local decision-making of agents can lead to sub-optimal solutions (required resources and operation allocations) and may not always yield globally optimal solutions. The work is limited in its detailed analysis of system performance metrics. Meanwhile, multi-objective optimization is crucial in RMS for balancing trade-offs (e.g., cost, time, resource allocation), so it is better handled by evolutionary algorithms as acknowledged in the current work.

[257] explores another MAS-based distributed method that addresses the design and optimization issues of context-aware workflow management systems for cyber-physical system (CPS) in IoT-enabled manufacturing environment (as RMS). The research focuses on the complexities of modeling and managing CPS operations. This approach uses discrete timed Petri nets (DTPN) to model the cyber world of the entities in the system and MAS architecture. Adopting discrete-valued firing time mean that transitions can only fire at specified discrete time intervals (e.g., integers or multiples of a base time unit). This specification can model digital systems and scenarios where time is naturally quantized such as computer simulations and digital control systems. Using discrete-valued firing time simplifies the modeling and analysis especially in computational models but may be less precise. Moreover, adopting Real-valued firing time in Petri nets models allow for continuous-time modeling. This specification is useful in systems where the precision of time is critical for optimizing processes that depend on exact timing. The current work focuses on fine-grained timing adjustments and ensuring the desired accuracy and optimal performance in the RMS to be studied.

In this contribution [258], we are interested to achieve the combination of "*Petri Nets*" and "*Genetic Algorithms*" by proposing a new formalism. The original idea involves extending reconfigurable object nets (RONs) [67] to deal with evolution (mutation and crossover). The versatility and effectiveness of Reconfigurable Object Nets (RONs) in various applications were presented in several researches. The works [259], [260], [261], [262] highlight RONs' theoretical underpinnings, practical implementations, and comprehensive frameworks for managing dynamic protocols. [259] presents foundational research on RONs, emphasizing their theoretical aspects and potential for managing dynamic protocol changes. [260] discusses the practical applications of RONs in distributed systems and it demonstrates how RONs can effectively manage complex interactions and maintain system integrity amidst evolving demands. [261] outlines a systematic approach to leveraging RONs for reconfiguring communication protocols, thereby accommodating new requirements and technological advancements. [261] proposes a framework that emphasizes the scalability and adaptability of RONs in handling the evolution of protocols in dynamic environments. Finally, [262] illustrates how RONs can be utilized to manage distributed interactions efficiently, ensuring that the system can cope with changes and continue functioning optimally.

In this approach, the new centralized extension called **Genetic RONs** (Gen-RONs) introduces mutation and crossover into PNs and provides them with a mathematical foundation based on graph transformation theory. Thus, the reproduction of generations using the mutation and crossover of Gen-RONs defines the reconfiguration process in RMSs. Hence, the Gen-RONs formalism integrates modeling/analysis capabilities of PNs with the optimization abilities offered by genetic algorithms. This combination effectively addresses (i) the scheduling problem by modeling the RMS configurations as Petri nets, unlike other approaches that rely on complex encoding/decoding procedures, (ii) the reconfiguration problem using the RONs formalism enriched with reconfiguration rules to ensure preservation of "required properties" from the initial RMS configurations.

This contribution study extends our work of [263]. We introduce a new algorithm based on the *inorder traversal rank* of the binary tree structure in the population initialization step to prevent deadlock. This ensures the generations of new Petri nets without deadlock. Additionally, the Taguchi's method [264] is applied to Gen-RONs to enhance solutions quality in the subsequent generations by tuning the parameters of the genetic algorithm.

To illustrate the contribution of the present chapter, a reconfigurable manufacturing system inspired by the case study reported in [217] is proposed to optimize the conflicting criteria of cost and completion time for system configurations. Additionally, the performance of the proposed approach is evaluated through extensive experiments utilizing Taguchi's method, assessing both convergence and diversity metrics. Comparative results between NSGA-II and Gen-RONs demonstrate that the Gen-RONs algorithm generates a greater number of solutions (configurations) than NSGA-II, achieving significant improvements in both completion time and cost. Importantly, the solutions obtained are deadlock-free and safe, preserving these qualitative properties from the initial configurations. This enhancement is attributed to the novel definition of genetic operators introduced by the proposed approach.

To the best of current knowledge, no existing research has applied reconfigurable object nets (RONs) to optimize of multi-criteria problems in RMSs using these methods. The following points highlight the originality of this contribution.

- We propose a new formalism that combines the RONs and GAs with a deadlock-free approach.
- The reconfiguration process is modeled by applying RONs' rules to PNs, facilitated by the proposed mutation and crossover mechanisms of Gen-RONs. This innovative approach ensures the preservation of the good properties after each reconfiguration, leveraging the robustness of the RONs formalism.

4.3 Basic concepts for modeling and optimization

This section presents key elements used for the modeling and optimizing RMSs.

4.3.1 Concepts for RMSs' modeling

The proposed Gen-RONs represent an innovative formalism derived from reconfigurable object nets (RONs). This subsection details the essential elements of RONs that facilitate the effective modeling of RMSs.

4.3.1.1 Reconfigurable Object Nets

RONs formalism was first introduced in [67] as an extension of Petri nets. RONs aim to provide a formal framework for the specification and simulation of reconfigurable systems. This formalism exploits graph transformation theories to implement reconfiguration over Petri nets. The basic characteristics of RONs is to provide two new types of tokens, which are "nets" and "rules". Thus, places of a RON can contain either net-tokens or rule-tokens. Net-tokens (called also object-nets) are represented as classical place/transition nets (P/T nets) and rule-tokens formalize the reconfiguration of P/T nets as a graph transformation process. Following this description, two levels are defined in an RON, the inside level (net-tokens: a set of P/T nets) and the outside level (the system net or the RON). The dynamic of the RONs is defined, basically, through three kinds of transitions: (i) *Fire transition* which changes the marking of a net-token (P/T net) by firing some transitions in this net-token, (ii) *Transform transition* which takes a rule-token and a net-token to yield a new net-token (the result of applying the rule onto the net-token), and (iii) *Ordinary transition* which acts as normal transitions in Petri nets and changes the location of a net-token from one place to another in the RON. Figure 4.1 shows an example of modeling a simple system by RONs formalism. In this example, there are four

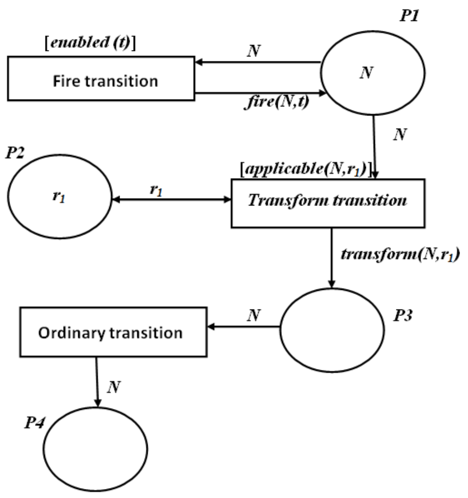


Figure 4.1: RON model of a simple system.

places in the RON model ($P1$, $P2$, $P3$, $P4$). $P1$, $P3$ and $P4$ are three token-net places and $P2$ is a token-rule place. Fire transition takes net N as parameter and updates the marking of N by firing t (if this last one is enabled) i.e., respecting the guard [$enabled(t)$ = true]. Once t is fired, a new net is produced by computing the function $fire(N, t)$. *Transform transition* takes net N and rule r_1 as parameters, then, it applies this rule to transform N respecting the guard $[applicable(N, r_1)]$. Thus a new net with a new structure is produced using function $transform(N, r1)$. Ordinary transition takes as parameter net N and changes its location from $P3$ to $P4$.

As previously mentioned, the reconfiguration of the structure in RONs pertains exclusively to the object nets (i.e., net-tokens at the inner level), meaning the overall structure of the net at the system level remains unchanged. This reconfiguration is inspired by "graph transformation techniques" [110]. Two primary graph transformation operators are utilized in Petri nets: (i) Union (referred to as the single push-out) and (ii) Transformation (known as the double push-out). The union operator takes two P/T nets N_1 and N_2, to produces another P/T net N_3. Conversely, the Transformation operator takes one P/T net N_1 and yields another net N_2. Basic definitions related to P/T nets and morphisms over P/T nets, which are essential for the union and transformation operators, are presented below.

4.3.1.2 Place/Transition Nets (P/T nets)

A place/transition net (P/T net) is a quadruplet $(T, P, Pre, Post)$, where:

- T is a non-empty finite set $\{t_1, \ldots, t_n\}$ of n transitions.
- P is a non-empty finite set $\{p_1, \ldots, p_m\}$ of m places. In the following paragraphs, $P^{\oplus}$ denotes the set of finite multi-sets over set P. If w is an element in $P^{\oplus}$, then it can be written as: $w = \Sigma_{p \in P} \lambda_p \times p$, such that λ_p is a natural number $\lambda_p \in \mathbb{N}$. It is also possible to consider w as a function, thus $w : P \to \mathbb{N}$.
- Pre (i.e., pre-domain) and $Post$ (i.e., post-domain) define respectively two mapping functions as follows. $Pre, Post : T \to P^{\oplus}$.

4.3.1.3 Morphisms over P/T nets

Consider two P/T nets defined as $N_1 = (T_1, P_1, Pre_1, Post_1)$ and $N_2 = (T_2, P_2, Pre_2, Post_2)$. A morphism f relating the nets N_1 and N_2 is defined as a function $f : N_1 \to N_2$. This morphism can be expressed as $f = (f_T, f_p)$, where $f_T : T_1 \to T_2$ maps transitions to transitions, and $f_P : P_1 \to P_2$ maps places to places. The two mappings f_T and f_P must satisfy the following two conditions.

- $Pre_2 \circ f_T = f_P^{\oplus} \circ Pre_1$
- $Post_2 \circ f_T = f_P^{\oplus} \circ Post_1$

Figure 4.2 [67] summarizes the previous concepts.

$$\begin{tikzcd}
\mathcal{T}_1 \arrow[r, shift left, "pre_1"] \arrow[r, shift right, "post_1"'] \arrow[d, "f_T"] & \mathcal{P}^{\oplus}{}_1 \arrow[d, "f_P^{\oplus}"] \\
\mathcal{T}_2 \arrow[r, shift left, "pre_2"] \arrow[r, shift right, "post_2"'] & \mathcal{P}^{\oplus}{}_2
\end{tikzcd}$$

Figure 4.2: Morphisms on P/T nets.

4.3.2 Concepts for RMSs' optimization

This subsection introduces key elements related to optimization within the proposed formalism, highlighting their significance and application.

4.3.2.1 Tuning Parameters using Taguchi's Method

The quality of solutions obtained through multi-objective evolutionary algorithms (MOEAs) depends on the input parameters of genetic algorithms. We employ Taguchi's method [264] to tune these parameters. The primary objective of Taguchi's method is to design experiments efficiently. This method uses orthogonal arrays to select combinations of levels (in the current approach, defining new values for each parameter is considered a level) for the input design variables (in our case, the variables are the GA parameters) for each experiment. By employing this method, we can gather comprehensive information about all factors influencing performance parameter in a minimal number of experiments. For result analysis, Taguchi's method applies two approaches: (i) variance analysis for experiments that repeat once, (ii) signal to noise ratio (S/N where S is controllable factors and N is noise factors) for experiments with multiple runs. The latter approach focusing on the signal to noise ratio is emphasized due to the multiple runs employed in this method, aimed at achieving best solutions. Overall, the application of Taguchi's method in this context not only facilitates effective parameter tuning but also contributes significantly to the robustness and quality of the solutions generated.

4.3.2.2 Genetic Algorithm Parameters: Choice and Discussion

Genetic algorithms, including NSGA-II [136], involve several parameters that significantly influence their effectiveness. The most commonly studied parameters are *population size*, *crossover rate*, *mutation rate* and *number of generations*. Population size and the number of generations impact the diversity of solutions and the algorithm's convergence, respectively. Larger values for these two parameters generally ensure better convergence while maintaining a diverse set of solutions. It is also essential to consider the rates of crossover and mutation, as these determine whether a chromosome will undergo crossover and/or mutation operations. The values assigned to these rates are typically drawn from the literature where the common practice is to assign a high value to the crossover rate and a low value to the mutation rate.

4.3.2.3 Performance Metrics

Several evolutionary algorithms have been proposed over the last few decades. Consequently, many performance metrics have been defined to compare these algorithms by measuring their effectiveness, efficiency and the quality of the solutions set produced. The three principal aspects that define the performance metrics for solutions obtained through evolutionary algorithms are [265]: (i) the convergence, (ii) the diversity and (iii) the number of solutions. Based on these aspects, we choose to evaluate our algorithm and compare it with NSGA-II using the coverage and spacing metrics which have been used also in [225]. Additionally, they will compare the algorithms in terms of the solutions' number. This comprehensive evaluation aims to provide a clearer understanding of the strengths and weaknesses of the proposed algorithm in relation to established benchmarks.

The coverage metric (CM) differentiates two Pareto fronts generated by two algorithms. It represents the rate of solutions obtained from one algorithm that are not dominated by any solutions obtained from the other algorithm. This metric is formally presented in Equation 4.1.

$$CM(P_i) = \frac{\|P_i - \{X \in P_i / \exists Y \in P : Y > X\}\|}{\|P_i\|} \tag{4.1}$$

where: $i = 1, 2$ therefore P_1, P_2 are two different sets of non-dominated solutions obtained by two different algorithms; $P = P_1 \cup P_2$; Y, X are two non-dominated solutions; $Y > X$ means

Y dominates X. P_i with CM closer to 1 is considered a better set of solutions. This metric provides a measure of the algorithm's convergence.

Example: Let P_1 and P_2 denote the Pareto front sets of algorithm 1 and algorithm 2, respectively, with $\|P_1\| = 5$ and $\|P_2\| = 7$. It is assumed that three solutions in P_2 dominate four solutions in P_1, while only one solution in P_1 dominates two solutions in P_2. Using Equation 4.1, it is found that $CM(P_1) = 0.2$ and $CM(P_2) = 0.71$. Since $CM(P_2)$ is closer to 1, P_2 is considered the better set of solutions. Thus, algorithm 2 performs better than algorithm 1 in terms of the better solutions set.

The spacing metric (SP) measures the well distribution of solutions in the non-dominated sets produced by the multi-objective evolutionary algorithms. This metric is presented in Equation 4.2. It measures the distance variance of neighboring vectors within the non-dominated sets.

$$SP = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{d} - d_i)^2} \tag{4.2}$$

such that d_i is computed as presented in Equation 4.3

$$d_i = min_j(\|f_1(x_i) - f_1(x_j\| + \|f_2(x_i) - f_2(x_j)\|) \tag{4.3}$$

where: n is the number of non-nominated solutions found so far; $\bar{d} = \sum_{i=1}^{n}(\frac{d_i}{n})$; f_1 and f_2 are two objective functions; $i, j = 1, 2, 3.....,n$. A smaller value of this metric indicates a better distribution of the solutions.

4.4 Proposed approach

In this section, we introduce the newly proposed formalism Gen-RONs. Subsection 4.4.1 begins with a comprehensive overview of this formalism. Subsection 4.4.2 outlines the problem description, followed by Subsection 4.4.3, which delves into the problem formulation. Finally, Subsection 4.4.4 elaborates on the proposed approach providing a detailed description and formalization of both the mutation and crossover operators.

4.4.1 Motivation

RMSs must be capable of making rapid changes at both hardware and software levels at runtime ensuring that system components do not encounter deadlocks. These requirements are for essential adjusting production capacity and features as well as for avoiding or addressing failures. However, any issues in such systems can lead to serious bugs.

To address these challenges, design of RMSs must consider two important aspects: the properties of configurations and the reconfiguration process itself. Assessing configuration properties allows for the management of critical aspects offline during the design phase such as deadlock freedom, safety, reliability, performance, cost, etc. On the other hand, focusing on the reconfiguration process involves determining optimal configurations in terms of several criteria. To achieve these objectives, the authors propose a new formalism Gen-RONs for specifying, analyzing and optimizing RMSs ensuring deadlock-free and safe solutions (configurations). This formalism uses reconfigurable object nets (RONs) to model RMS configurations as P/T nets with the reconfiguration process formulated as transformations over these nets. In addition to specifying RMSs, Gen-RONs optimize RMSs in terms of the required criteria using genetic algorithms. Reconfigurations of RMSs, specified as RONs reconfigurations, are considered mutation and crossover operators within the Gen-RONs framework. Thus, mutation and crossover are defined to preserve required properties, such as deadlock freedom, liveness and safety, of the initial configurations. Furthermore, the authors integrate the technique of inorder traversal rank [266] in the first step of the genetic algorithm, to prevent deadlocks during the generation of the initial population. Consequently, Gen-RONs ensure deadlock-free and safe configurations throughout the reproduction process. In addition to these innovations, we anticipate further improvements in the number of optimized configurations obtained through the encoding technique using P/T nets.

4.4.2 Problem Description

Reconfigurable manufacturing systems are characterized by the use of reconfigurable machines. These machines change their configurations depending on the tools available and the specifications of the product design. The authors assume that each configuration provides different degrees of freedom along the three-dimensional axes *x*, *y*, and z. A product type is considered a set of parts. Each part has a set of operations that must be performed while respecting precedence relationship among them. Therefore, the authors aim to determine the appropriate scheduling of the different operations for a product type within RMSs focusing on properties such as reliability, deadlock-freedom, cost-effectiveness and optimal scheduling.

4.4.3 Problem Formulation

This subsection presents the problem formulation by exploring two critical phases: the modeling phase and the optimization phase.

4.4.3.1 RMS Modeling

The modeling phase enhances the formal definition of P/T nets by incorporating critical information related to time and cost to effectively model RMSs. The new extension is called labeled-P/T nets (L-P/T nets). Formally, a labeled-P/T net LN = (N, LT, LC), where:

- $N = (P, T, Pre, Post)$ is a P/T net.
- LT and LC are two label functions that associate time and cost to each transition $t \in T$ as follows. $LT, LC : T \to \mathbb{R}^+$.

The reconfiguration in RMSs will not change the labels of the transitions.

L-P/T nets are employed to model the chromosomes representing the initial configurations of the population. Consequently, a solution is referred to as an L-P/T chromosome. An L-P/T chromosome consists of a set of L-P/T genes. Each L-P/T gene models an operation to be performed in an RMS with its required machine. Each operation is modeled by a transition op_i where i is the operation number. Each machine mop_i is initially represented by a place in the abstract model, while its behavior is modeled by another L-P/T net. Other places and transitions are included such as the start places (P_{start}), the transfer transitions (tr), after-operation places (p_i) and before-operation places (bop_i). The start and after-operation places serve as gluing places to connect L-P/T genes, forming the complete L-P/T chromosome. Figure 4.3 shows an example of an L-P/T gene for modeling an RMS operation op_1.

Running example. Consider an RMS with two machines M_1 (equipped with two tools t_1 and t_2) and M_2 (equipped with one tool t_3). Each tool has a specific configuration. A product

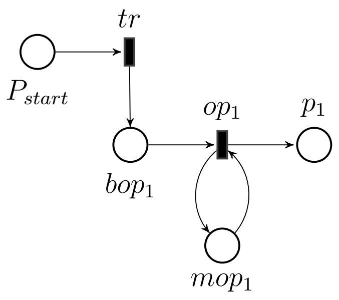


Figure 4.3: The L-P/T gene of the operation 1.

pr_1 with one part is to be produced, requiring three operations: op_1, op_2 and op_3. The operation op_1 can be performed on M_1 with tool t_1, op_2 can be performed on M_2 with tool t_3 and op_3 can be performed on M_1 with tool t_1. It is assumed that op_1 must be performed first followed by op_3 and finally op_2. Figure 4.4 illustrates the complete abstract RMS configuration model.

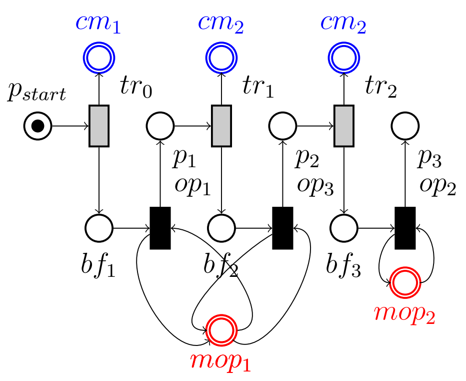


Figure 4.4: An abstract L-P/T net of the example.

To enhance the clarity and readability of the L-P/T net model for the RMS configuration, it is designed in a modular manner. Specifically, the entire model is divided into an abstract template and a set of L-P/T nets that model the machines. The abstract template model and the machine models are connected through a set of communication places, denoted by double circles in the models, as depicted in Figures 4.4 and 4.5.

In Figure 4.4, place P_{start} represents the input of raw material needed to produce product Pr_1, while place p_3 receives the final product as the output of the RMS. The three operations are modeled by transitions (filled black color) op_1, op_2 and op_3. These operations are realized through two configurations (machines with specific tools) abstractly modeled by places mop_1, and mop_2. These places, denoted by red double circles, link the template to the L-P/T nets of machines and are defined based on the chosen configuration of the RMS. Each operation is initiated at an appropriate time by a specific command. The commands that launch the operations are modeled by places cm_1, cm_2 and cm_3. These command places, represented as blue double circles, also connect the template to the L-P/T nets of the machines. Finally, transitions (filled in grey color) tr_0, tr_1 and tr_2 model the transfer of product from one operation to another during runtime.

In Figure 4.5, place M_1_idle indicates that machine M_1 is idle. Places tl_1 and tl_2 denote that the two tools are free. Transitions get_tl_1 and get_tl_2 are used to acquire one of the two tools, respectively. These transitions are conditioned by the presence of the machine in an idle state, a free tool, and a command in one of the places cm_tl_1 and cm_tl_2. Places cm_tl_1 and cm_tl_2

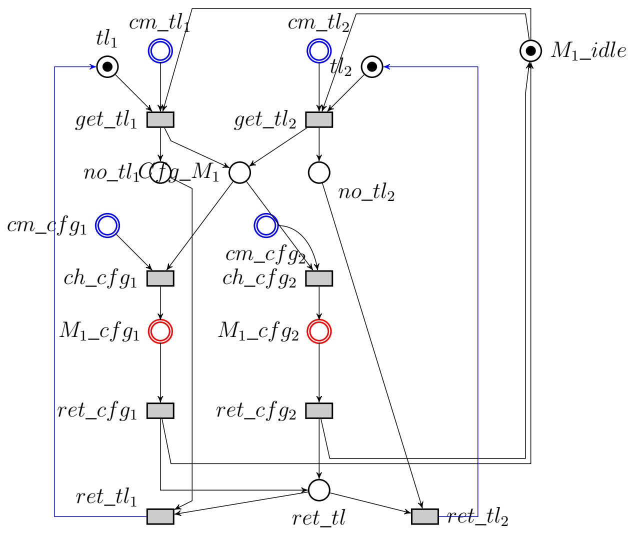


Figure 4.5: The L-P/T net of machine M_1.

are communication places that will be marked by a token when one of the communication places (cm_1, cm_2, cm_3) in the abstract model (Figure 4.4) is marked. Once the tool is acquired, places no_t_1 or no_t_2 will be marked as "no more free". Then, machine M_1 will be in one of its configurations $M_1\_cfg_1$ or $M_1\_cfg_2$, depending on the available command in cm_cfg_1 or cm_cfg_2. Places $M_1\_cfg_1$ and $M_1\_cfg_2$ are communication places that link the M_1 model with the abstract model. When place $M_1\_cfg_1$ or $M_1\_cfg_2$ is marked, place mop_1 (in Figure 4.4) will be marked, triggering one of the operations op_1 or op_3. Places cm_cfg_1 and cm_cfg_2 are communication places that will be marked when one of the places cm_1, cm_2 or cm_3 in Figure 4.4 is marked. From any of its configurations (i.e., $M_1\_cfg_1$, $M_1\_cfg_2$), M_1 can return to its idle state when either transition ret_cfg_1 or ret_cfg_2 is fired. The tool becomes free after firing either transition ret_tl_1 or ret_tl_2. Figure 4.5 models machine M_1. M_2 is modeled in a similar manner.

4.4.3.2 RMS Optimization

Significant related works on RMSs optimization focus on two key criteria: cost and/or the completion time, as demonstrated in studies such as [249], [250], [225], [222] and [267]. This study addresses the challenge of identifying optimal RMS configurations by simultaneously minimizing both completion time and cost, which are often in conflict. The notations used in this context are defined in Table 4.1.

The objective functions are : $Min\ f_{time}$ and $Min\ f_{cost}$ where f_{time} and f_{cost} denote the total time and total cost, respectively.

- **Total cost:**

  Total cost f_{cost} is the sum of the machining usage cost (MUC), the configuration changing cost (CCC), the tool usage cost (TUC) and the tool changing cost (TCC).

$$f_{cost} = MUC + CCC + TUC + TCC, \tag{4.4}$$

where:

Table 4.1: Notation used in RMS optimization.

Symbol/Variable	Meaning
$Part_{nbr}$	the parts number in a product
OP_{nbr}	the operations number
OPN_i	the operations number of part i
m	the machines number in an RMS
NC_i	the configurations number of the i^{th} machine
$MC[i][c]$	the matrix of available configuration c for machine i
$MT[i][t]$	the matrix of available tool t for machine i
$Cop[c][op]$	the matrix of the configuration c of the operation op
$Top[t][op]$	the matrix of the required tool t for performing operation op
$C[c][6]$	the matrix of the three-dimensional axis configurations x, y and z (+x, -x, +y, -y, +z, -z)
$PR[op][op]$	the precedence relationship between the operations
MC_i	the utilization cost of machine i
$TC[t]$	the tool t utilization cost
$ChM[i][j]$	the matrix of the costs of changing machine i to machine j
$ChC_m[i][j]$	the matrix of the costs of changing configuration i to configuration j for machine m
$ChT_m[i][j]$	the matrix of the costs of changing tool i to tool j for machine m
$ChCT[i][j]$	the matrix of the time of changing configuration i to configuration j for each machine
$ChTT[i][j]$	the matrix of the time of changing tool i to tool j for each machine
$ChTM[i][j]$	the matrix of changing machine i to machine j
$PT[op]$	the processing time of operation op
T_l^i	the tool l used to accomplish the operation i
$T_{l'}^{nextOP}$	the tool l' used to accomplish the operation to be performed after the operation i
$ChT_{l,l'}$	the cost of changing the tool l to the tool l' in the same machine

– MUC depends on $PT[OP]$ of $Part_j$ for a specific machine i.

$$MUC = \sum_{i=0}^{OPN-1} MC_i \times PT_{Part_j}[OP_i]. \tag{4.5}$$

– CCC depends on the current operation i and the next operation $nextOp$ and their machine configurations.

$$CCC = \sum_{i=1}^{OPN} \Phi(Cop[c][i], Cop[c'][nextOp]) \times ChC[c, c'], \tag{4.6}$$

where;

$$\Phi(Cop[c][i], Cop[c'][nextOp]) = \begin{cases} 0, & \text{if } Cop[c][i] = Cop[c'][nextOp] \\ 1, & \text{otherwise} \end{cases} \tag{4.7}$$

- TUC depends on the tool used and the processing time PT of part $Part_j$.

$$TUC = \sum_{i=0}^{OPN-1} TC[i] \times PT_{Part_j}[OP_i], \tag{4.8}$$

- TCC depends on the current operation i and the next operation $nextOp$ and their tool used in the same machine.

$$TCC = \sum_{i=1}^{OPN-1} \Phi(T_l^i, T_{l'}^{nextOp}) \times ChT_{l,l'}, \tag{4.9}$$

where;

$$\Phi(T_l^i, T_{l'}^{nextOp}) = \begin{cases} 0, & \text{if } T_l^i = T_{l'}^{nextOp} \\ 1, & \text{otherwise} \end{cases} \tag{4.10}$$

- **Total time:**

Total time f_{time} is the sum of the processing time of a specific operation (PT), the time for tool changeover (TCT) and the time for the configuration changing (CCT).

$$f_{time} = PT + TCT + CCT, \tag{4.11}$$

where:

- PT depends on the machine, its configuration and the type of the operation.

$$PT = \sum_{i=1}^{OPN_{partk}} PT_{MC_i}, \tag{4.12}$$

where;

OPN_{partk} is the total number of the operations in each part and PT_{MC_i} is the time required for the machine m with the configuration c to perform the operation i of the part k.

- TCT depends on the type of the operation to be performed.

$$TCT = \sum_{i=1}^{OPN-1} ChTT[T_l^i, T_{l'}^{nextOp}], \tag{4.13}$$

- CCT depends on the machine and its current and next configurations.

$$CCT = \sum_{i=1}^{OPN-1} ChCT(Cop[c][i], Cop[c'][nextOp]), \tag{4.14}$$

4.4.4 Genetic Reconfigurable Object Nets (Gen-RONs)

In this section, the proposed formalism, Gen-RONs, is introduced, providing new formal definitions for the genetic algorithm operators of crossover and mutation. Furthermore, an important characteristic achieved by this formalism is also discussed.

4.4.4.1 Formalization of Mutation Operator

In genetic algorithms, mutation randomly alters a chromosome by updating one or more genes within it. When considering a chromosome as an L-P/T net, the genes represent the fundamental components of the net, including places, transitions, arcs, places' marking and arcs' labels. Applying mutation to an L-P/T net involves reconfiguring its structure by updating components. However, this mutation must adhere to a set of syntactic constraints to ensure that the resulting chromosome remains a valid L-P/T net (e.g., each arc must have a source and a target nodes, and the source and target nodes of an arc must not be of the same type, etc.). In Gen-RONs, the mutation operator (denoted as Mut_{GenRON} can modify various features of the L-P/T net nodes, including names, layout, places' marking. The mutation operation is treated as a transformation of the L-P/T net (i.e., double push-out). The double push-out over L-P/T nets combines two unions (i.e., two single push-outs). Let $L, I, R,$ and C represent four L-P/T nets. A transformation is defined as an operator f: $f : N_1 \to N_2$ that transforms L-P/T net N_1 into L-P/T net N_2 using the rule $r = (L, I, R)$ and match $m : L \to N_1$. Figure 4.6 illustrates a transformation operator constructed using two single push-outs.

$$\begin{tikzcd}
\mathcal{L} \arrow[d, "m"] & \mathcal{I} \arrow[l, "k_1"'] \arrow[r, "k_2"] \arrow[d, "c"] & \mathcal{R} \arrow[d, "n"] \\
\mathcal{N}_1 & \mathcal{C} \arrow[l] \arrow[r] & \mathcal{N}_2
\end{tikzcd}$$

Figure 4.6: Double Pushout.

In Figure 4.6, k_1, k_2, m, c, and n are morphisms. Net C is the context of the transformation. C must satisfy the following conditions.

- $T_C = (T_1 \setminus m_T(T_L)) \cup m_T(k_{1_T}(T_I))$;
- $P_C = (P_1 \setminus m_P(P_L)) \cup m_P(k_{1_P}(P_I))$;
- $Pre_C = Pre_{1|T_C}$ (Pre_C is the subset of Pre_1 which concerns only set of transitions T_C);
- $Post_C = Post_{1|T_C}$ ($Post_C$ is the subset of $Post_1$ which concerns only the set of transitions T_C).

Example: an illustrative example is presented in Figures 4.7 and 4.8. Figure 4.7 shows the L-P/T chromosome N_1, while its mutated version, resulting in L-P/T chromosome N_2, is depicted in Figure 4.8.

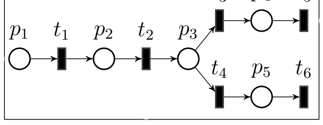


Figure 4.7: The L-P/T chromosome N_1.

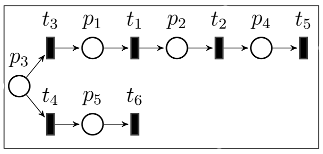


Figure 4.8: The L-P/T chromosome N_2 after the mutation of N_1.

The above mutation must be justified within the framework of L-P/T nets transformation theory. This mutation is proposed to be defined as a Double-Push Out (DPO) in the graph transformation theory. To confirm the validity of the mutation, it is necessary to demonstrate the existence of a DPO, referred to as $DPO1$. Figure 4.9 illustrates an existing production rule $p = (L, I, R)$ used in $DPO1$, while the morphisms from L to L-P/T chromosome N_1 and from R to L-P/T chromosome N_2 are graphically represented in Figures 4.10 and 4.11, respectively. Finally, Figure 4.12 presents the context of $DPO1$.

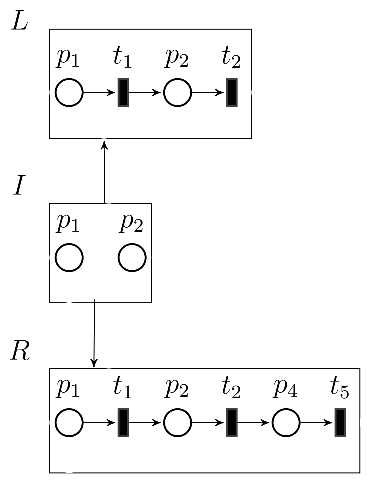


Figure 4.9: The production rule of the DPO1.

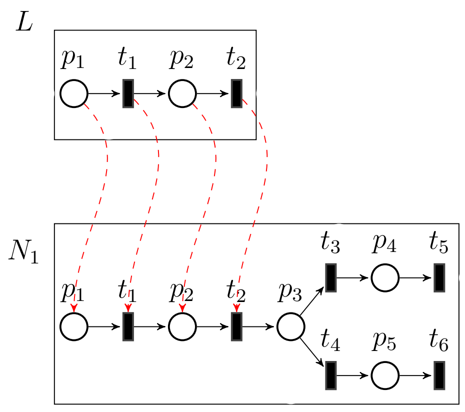
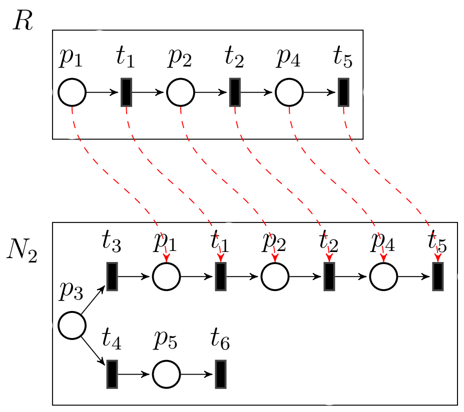


Figure 4.10: The injective morphism from L to the L-P/T chromosome N_1.



R
p_1 t_1 p_2 t_2 p_4 t_5
N_2
t_3 p_1 t_1 p_2 t_2 p_4 t_5
p_3
t_4 p_5 t_6

Figure 4.11: The injective morphism from R to the "L-P/T chromosome after mutation" N_2.

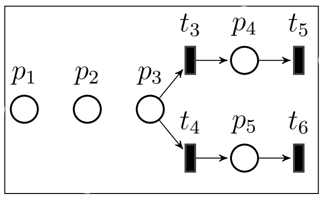


Figure 4.12: The context of the DPO1.

This proof validates that the mutation $N_1 \rightarrow N_2$ adheres to the principles of L-P/T nets transformation theory.

4.4.4.2 Formalization of Crossover Operator

In Gen-RONs, the crossover operation is defined as a union constructor. The crossover operator, denoted as $Cros_{GenRON}$, requires two parent L-P/T nets chromosomes, Net_1 and Net_2, and produces two new offspring, Net_3 and Net_4. The construction of the offspring proceeds as follows: first, two subnets Net'_1, Net''_1 are extracted from Net_1, and two subnets Net'_2 and Net''_2, are extracted from Net_2. Subsequently, Net'_1 and Net'_2 are glued using the union operation to form chromosome Net_3. Finally, Net''_1 and Net''_2 are glued to create chromosome Net_4. Applying crossover to an L-P/T net involves reconfiguring its structure by adding, deleting or updating components. Importantly, resulting L-P/T net chromosomes, Net_3 and Net_4, must adhere to the transformation graph constraints, ensuring that both Net_3 and Net_4 are DPOs of one of the parent chromosomes, Net_1 or Net_2.

The union is a binary operator constructed over morphisms and L-P/T nets. Consider the following three L-P/T nets: $N_1 = (T_1, P_1, Pre_1, Post_1)$, $N_2 = (T_2, P_2, Pre_2, Post_2)$, and $I = (T_0, P_0, Pre_0, Post_0)$. Furthermore, let $f : I \to N_1$ and $g : I \to N_2$ represent two morphisms. Net I serves as a common interface between N_1 and N_2. The union operator of N_1 and N_2 produces a new net $N = (T, P, Pre, Post)$, for which two morphisms $f' : N_1 \to N$ and $g' : N_2 \to N$ exist. The notation $N = N_1 +_I N_2$ indicates that the operator $+_I$ represents the union, referred to as **a single pushout** construction or the **gluing** operator, as illustrated in Figure 4.13.

$$\begin{tikzcd}
I \arrow[r, "f"] \arrow[d, "g"'] & N_1 \arrow[d, "g'"] \\
N_2 \arrow[r, "f'"'] & N
\end{tikzcd}$$

Figure 4.13: Union of L-P/T nets.

Indeed, union net N is built as follows:

- $T = T_1 +_{T_0} T_2$. T is said a disjoint union of the two sets T_1 and T_2, where the authors glue together all couples of transitions: $(f_T(t), g_T(t))$ such that $t \in T_0$;
- $P = P_1 +_{P_0} P_2$. P is said a disjoint union of the two sets P_1 and P_2, where the authors glue together all couples of places: $(f_P(p), g_P(p))$ such that $p \in P_0$;
- $Pre(t) = Pre_1(t_1)$ (*resp*, $Pre_2(t_2)$) if $g'_T(t_1) = t$ (*resp*, if $f'_T(t_2) = t$);
- $Post(t) = Post_1(t_1)$ (*resp*, $Post_2(t_2)$) if $g'_T(t_1) = t$ (*resp*, if $f'_T(t_2) = t$).

Example: Consider the example of crossover where N_1 and N_2 are the two L-P/T chromosomes depicted in Figure 4.14. Within the red boxes, two subnets N'_1 and N'_2 are highlighted, derived from N_1 and N_2, respectively. The subnets outside the red boxes represent the remaining components, N''_1 and N''_2, of N_1 and N_2, respectively. Figure 4.15 illustrates offspring, N_3 and N_4, which are the resulting L-P/T chromosomes derived from the two parent nets N_1 and N_2.

The crossover depicted in the union part of Figure 4.16 produces L-P/T offspring N_3 (derived from N'_1 and N'_2) and employs a simple interface I, which consists only of one place p. This place is mapped to p_3 in N'_1 (and to p_8 in N'_2), resulting in $k(p) = p_3$ and $k'(p) = p_8$. Morphisms $l : N'_1 \to N_3$ and $l' : N'_2 \to N_3$ are defined as follows: $l = \{(p_3, p_3), (p_4, p_4),$

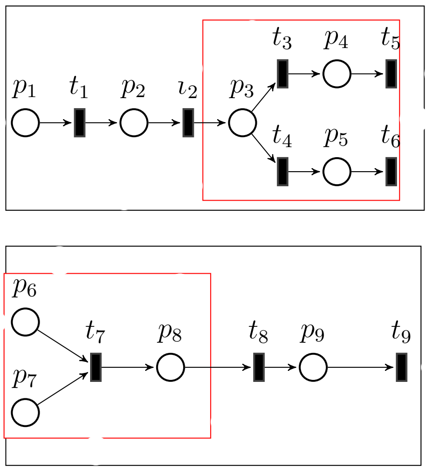


Figure 4.14: Two L-P/T chromosomes N_1 and N_2.

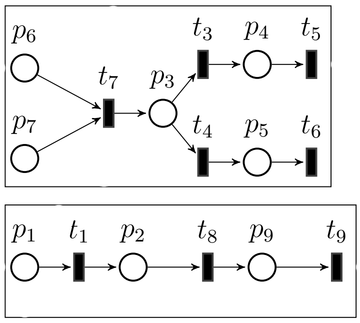


Figure 4.15: The L-P/T offspring chromosomes N_3 and N_4.

(p_5, p_5), (t_3, t_3), (t_4, t_4), (t_5, t_5), $(t_6, t_6)\}$ and $l' = \{(p_6, p_6), (p_7, p_7), (p_8, p_3), (t_7, t_7)\}$. Using the definition of the union, it can be expressed as $N_3 = N_1' +_I N_2'$. Informally, I serves as the disjoint union of N_1' and N_2' up to the interface I. Consequently, the places $l(p_3)$ and $l'(p_8)$ are glued as single place p_3 in N_3. Additionally, N_3 is computed using the double push-out to ensure it represents a valid transformation of either N_1 or N_2, similar to the mutation example discussed previously. Following the same method, the second offspring, N_4, can be generated using another union based on N_1'' and N_2''.

As illustrated in Figure 4.16, I represents an interface for the single push-out "SPO", facilitating the initial union of the two subnets N_1' and N_2' from N_1 and N_2, respectively, to produce the first offspring, N_3. Meanwhile, I' serves as the interface for the double push-out "DOP", confirming that N_3 is a valid transformation of N_1).

$$\begin{tikzcd}
\mathcal{I} \arrow[r, "k"] \arrow[d, "k'"] & \mathcal{N}'_1 \arrow[d, "l"] & & \\
\mathcal{N}'_2 \arrow[r, "l'"] & \mathcal{N}_3 & \mathcal{C} \arrow[l] \arrow[r] & \mathcal{N}_1 \\
& \mathcal{R} \arrow[u, "n"] & \mathcal{I}' \arrow[l, "k_2"'] \arrow[r, "k_1"] \arrow[u, "c"] & \mathcal{L} \arrow[u, "m"]
\end{tikzcd}$$

Figure 4.16: The first crossover produces the first offspring N_3: $N_3 = N_1' \otimes N_2'$ where (a) N_3 is DPO of N_1 and (b) N_3 is DPO of N_2

4.4.5 Properties Preservation through Crossover and Mutation Operators

As mentioned, Gen-RONs formalism is based on reconfigurable object nets [67], which utilize graph transformations applied to P/T nets. These transformations have been proved to preserve specific properties from the source P/T net to the resulting P/T net after transformation (see page 534, about achieved results in [110]). Consequently, the proposed formalism preserves the good properties. This important characteristic, on one hand, guarantees essential properties such as deadlock-freedom, liveness and safety. On the other hand, it optimizes and enhances the verification process. This is achieved by executing the verification process solely on the initial population, ensuring that the good properties remain verified throughout the production process using the proposed mutation and crossover. Therefore, if N generations are computed, then checking properties is required only for the first generation. The next $N-1$ generations do not require further checking, resulting in a verification efficiency gain of $(N-1)/N$. The authors focus on three fundamental properties: **liveness**, **safety** and **deadlock-freedom**. These properties are preserved when applying Petri nets' transformations to a P/T net or an L-P/T net. The following paragraphs provide both informal and formal definitions of live, safe and deadlock-free Petri nets.

Definition 4 *(Liveness), a liveness property imposes that it will be always possible for "good things" to occur in the future. For example, in RMSs a raw material must always be treated by the required machines.*

Definition 5 *(Safety), a safety property imposes that some "bad thing" will not occur during the systems execution. Safety includes several requirements like: mutual exclusion, deadlock freedom, etc. For example, in an RMS a failed product must never pass the test stage.*

Definition 6 *(Live Petri net), Formally, A live Petri net can be defined (as defined in [268]) as follows. Let (N, M_0) be a Petri net with its initial marking M_0. Let $R(M_0)$ be the set of all reachable markings from M_0. A Petri net N is live if all its transitions are live. A transition $t \in T$ is live iff $\forall M \in R(M0)$, $\exists M \in R(M)$ such that $M[t >$. Thus, transition t will eventually be enabled at some marking reachable from $M0$.*

Definition 7 *(Safe Petri net), as defined in [268], a Petri net N is k-bounded if the $M(p) < k$, $\forall p \in P$ and $\forall M \in R(M0)$. A Petri net N is safe if it is 1-bounded.*

Definition 8 *(Deadlock-free Petri net) as defined in [268], a marked Petri net $(N, M0)$ is deadlock-free iff $forall M \in R(M0)$ there is some transition $t \in T$ that is enabled. A deadlock has no enabled transitions.*

4.5 Gen-RONs based Genetic Algorithm

This section presents the proposed algorithms used in Gen-RONs and discusses their complexities. At this stage, a new formalism has been defined, enriched with the following capabilities: (i) modeling distributed systems with causality/concurrency relations among components, similar to ordinary Petri nets, and (ii) enabling reconfiguration through the restructuring of models via mutation and crossover operations typically employed in genetic algorithms.

4.5.1 Gen-RONs based NSGA-II principles

To elaborate on the evolution steps, Gen-RONs can leverage any existing genetic algorithm. The key impact of Gen-RONs lies in the encoding of chromosomes and the evolution operators. Specifically, Gen-RONs mutation and crossover operators are employed in the reproduction of offspring, leading to the proposal of a Gen-RONs-based genetic algorithm. This algorithm is based on the following principles.

- Chromosome encoding in Gen-RONs: The first step in implementing the genetic algorithm involves encoding, where the chromosomes are L-P/T nets. Thus, a solution is represented by a single L-P/T chromosome that represents the entire system. An L-P/T chromosome is viewed as a set of L-P/T genes. Each L-P/T gene models one operation to be performed in an RMS with the machine that completes this operation. Gluing places are utilized to connect these L-P/T genes.

- Initialization of the Gen-RONs based algorithm: The initial population is generated randomly. However, to prevent deadlocks within the configurations (i.e., L-P/T chromosomes), a technique based on the inorder traversal rank of the binary tree structure is used, as illustrated in Algorithm 5.

- Crossover and mutation operators: Mutation and crossover operators are denoted as Mut_{GenRON} and $Cros_{GenRON}$, respectively. They are applied after selecting a subset from the initial population. However, the processes may produce infeasible chromosomes. In this approach, an L-P/T gene (i.e., a sub L-P/T net within in an L-P/T chromosome) models a single operation. Consequently, a crossover between two L-P/T chromosomes may result in an RMS with an inappropriate set of operations or include an operation more times than necessary. To avoid these problems, a crossover between two parents N_1 and N_2 (i.e., two L-P/T nets) must respect the following constraints. Parents N_1 and N_2 are selected randomly. A sub L-P/T N_1' is randomly chosen from first parent N_1. Subsequently, a sub L-P/T N_2' is selected from N_2 and must include the same operation specified in L-P/T N_1'. Finally, the proposed crossover operator, along with its associated rate (crossover probability), is applied to the two selected L-P/T nets N_1 and N_2. In the mutation process, an L-P/T gene L is randomly selected from the L-P/T chromosome N_1. A mutation point is chosen while adhering to the constraints of the operations' precedence relationships. Finally, the proposed mutation operator is applied with the specified rate. Operators Mut_{GenRON} and $Cros_{GenRON}$ (formalized in Sections 4.4.4.1 and 4.4.4.2) use: (i) Algorithm 2 to verify morphisms between L-P/T nets, (ii) and Algorithm 3 to check for the existence of DPO between an offspring generated by $Cros_{GenRON}$ and its parents (N_1 or N_2), or to determine the existence of DPO between an L-P/T chromosome N and its mutant N' generated by Mut_{GenRON}. Thus, both crossover and mutation serve as transformations within the evolution process. Preserving the essential properties of *safety* and *liveness*. Finally, Algorithm 4 employs these two operators to define the evolution process across generations which will be integral to overall optimization outlined in Algorithm 1.

Algorithm 1 illustrates the previously discussed Gen-RONs steps integrated with NSGA-II algorithm [136]. NSGA-II is a widely used genetic algorithm for optimizing multi-objective problems. It is relies on non-dominated sorting, a crowded comparison operator and elitism-based selection. As a benchmark, NSGA-II has been referenced in numerous recent studies for comparative purposes, including works such as [267], [222], [269], [270], [271] and [272].

4.5.2 Entire Optimization Time Complexity

After randomly generating an initial population, Gen-RONs employs the technique of inorder traversal rank to avoid deadlocks, as implemented in Algorithm 5. Then, Gen-RONs performs optimization using a pre-existing genetic algorithm that incorporates the proposed mutation (Mut_{GenRON}) and crossover ($Cros_{GenRON}$). These operators utilize Algorithms 2

Algorithm 1 NSGA-II algorithm based on Gen-RONs.

```
 1: procedure NSGA-II_GEN-RONS
Require:
    $Pop = \{N_1, ..., N_l\}$                          ▷ a population of L-P/T chromosomes
    $max\_nb\_it$                                      ▷ the maximum number of iterations
    $PopSize$                                          ▷ the size of population
Ensure:
    $Pop = \{N_1, ..., N_l\}$                          ▷ a population with the best L-P/T chromosomes

 2:     $Pop \leftarrow initializePop(PopSize)$
 3:     $Pop \leftarrow DeadlockAvoidance(Pop)$
 4:     $Check\_liveness\_and\_safety(Pop)$
 5:     $Fronts \leftarrow FastNonDominatedSort(Pop)$
 6:     for each individual in Fronts do
 7:         CrowdingDistance-Computation(individual)
 8:     end for
 9:     $SortByCrowdedComparaisonOperator(Pop)$
10:     $Prt_{Pop} \leftarrow Select(Pop, PopSize)$
11:     $Children \leftarrow Evolution(Prt_{Pop})$
12:     $Pop \leftarrow Prt_{Pop} \bigcup Children$
13:     $i \leftarrow 0$                                 ▷ the number of current iterations
14:     while $i <= max\_nb\_it$ do
15:         $Fronts \leftarrow FastNonDominatedSort(Pop)$
16:         for each individual in Fronts do
17:             CrowdingDistance-Computation(individual)
18:         end for
19:         $SortByCrowdedComparaisonOperator(Pop)$
20:         $Prt_{Pop} \leftarrow Select(Pop, PopSize)$
21:         $Children \leftarrow Evolution(Prt_{Pop})$
22:         $Pop \leftarrow Prt_{Pop} \bigcup Children$
23:         $i \leftarrow i + 1$
24:     end while
25: end procedure
```

and 3 during the reproduction of offspring. Thus, the complexity of the entire Gen-RONs-based genetic algorithm is influenced by the complexities of the binary tree-based deadlock avoidance algorithm, the genetic algorithm used for optimization and the morphism and double-pushout algorithms. First, the complexity of the deadlock avoidance algorithm based on the inorder traversal of a binary tree structure is $O(n)$, where n is the number of nodes (i.e., the number of L-P/T genes in an L-P/T chromosome, which represents the number of operations to be performed in RMSs). Secondly, Gen-RONs employ NSGA-II as a GA; the overall complexity of NSGA-II is $O(M * N^2)$, where M is the number of objectives and N is the population size as mentioned in the original NSGA-II [136]. This complexity arises from the non-dominated sorting process (line 15 in Algorithm 1). Finally, the complexity of morphism algorithm is $O(o * N)$, where o is the number of L-P/T genes in an L-P/T chromosome (i.e., number of operations to be performed in an RMS) and N is the population size. For the DPO test algorithm, the complexity is $O(N)$ where N is the population size. Therefore, the overall time complexity of the proposed Gen-RONs-based NSGA-II still remains $O(M * N^2)$.

Algorithm 2 Morphism.

```
 1: procedure MORPHISM
Input:
    $N'$                                   ▷ an L-P/T gene
    $N$                                    ▷ an L-P/T chromosome
Output:
    $mor$ true or false                    ▷ if true, $mor$ is a morphism defined by: $mor : N' \to N$

 2:     $mor \leftarrow false$
 3:     for each $L - P/Tgene \in N$ do
 4:         if $N' \subseteq$ L-P/T gene then
 5:             $mor \leftarrow true$
 6:             break
 7:         end if
 8:     end for
 9: end procedure
```

Algorithm 3 Double Push-Out test.

```
 1: procedure DPO TEST
Input:
    $L$, $R$                               ▷ two L-P/T genes
    $N_1$, $N_2$                           ▷ two L-P/T chromosomes
Output:
    $DOP$ true or false                    ▷ if true, $N_2$ is a DPO of $N_1$: $N_1 \to N_2$

 2:     $DPO \leftarrow false$
 3:     $I \leftarrow L \cap R$                        ▷ $I$ is an interface
 4:     if morphism($I$, $L$) and morphism($I$,$R$) then
 5:         $C \leftarrow N_1 - (L \cup I)$            ▷ $C$ is the context
 6:         if morphism($I$, $C$) and morphism($C$, $N_1$) and morphism($C$, $N_2$) then
 7:             $DPO \leftarrow true$
 8:         else $DPO \leftarrow false$
 9:         end if
10:     end if
11: end procedure
```

Algorithm 4 Evolution process in Gen-RONs.

```
 1: procedure EVOULTION
Input:
    $Pop = \{N_1, ..., N_l\}$               ▷ Parent population
Output:
    $Children$                             ▷ Offspring

 2:     $Pop \leftarrow$ Apply $Mut_{GenRON}$ on $Pop$
 3:     $Children \leftarrow$ Apply $Cros_{GenRON}$ on $Pop$
 4: end procedure
```

Algorithm 5 Binary tree based deadlock avoidance.

```
 1: procedure DeadlockAvoidance
Input:
      $Population = \{L-P/Tchrom\_1, ..., L-P/Tchrom\_n\}$
Output:
      $Population$ of L-P/T chromosomes without deadlock
 2:     for each $L-P/Tchromosome$ in $Population$ do
 3:         Generate randomly $the\ L-P/Tchromosome$
 4:         $i \leftarrow 1$
 5:         while $i <= Nbr\_operations$ do
 6:             $label1 : root \leftarrow ith\_operation$
 7:             $label2 : leaf \leftarrow i+1th\_operation$
 8:             if precedence(root, leaf)==True then
 9:                 if $LeftChild(root) == None$ then
10:                     $Insert\_Left(leaf)$
11:                     $i \leftarrow i+1$
12:                 else if $LeftChild(root)$ != $None$ then
13:                     $root \leftarrow root\_LeftNodePoint$
14:                     $goto\ label2$
15:                 end if
16:             else if precedence(root, leaf)==False then
17:                 if $RightChild(root) == None$ then
18:                     $Insert\_Right(leaf)$
19:                     $i \leftarrow i+1$
20:                 else if $RightChild(root)$ != $None$ then
21:                     $root \leftarrow root\_RightNodePoint$
22:                     $goto\ label2$
23:                 end if
24:             end if
25:         end while
26:     end for
27: end procedure
```

4.6 Experimentation

This section presents the application of Gen-RONs in a case study of an RMS, inspired from [217]. The results are computed using a prototype that implements the Gen-RONs-based NSGA-II algorithm. More information about this tool and the case study can be found at `https://kahloul2006.wixsite.com/laid-kahloul/projects`.

4.6.1 RMS Description

Consider an RMS comprising three reconfigurable machines (M_1, M_2 and M_3) capable of achieving various functionalities based on their configurations. The operations performed by each machine depend on its available tools. Consequently, at any given time, a machine operates in a specific configuration with a defined set of available tools. The machine's configuration determines the degrees of freedom (motion along x, y and z axes) for each tool, known as the tool approach directions (TAD). In this case study, the RMS produces a single product consisting of three independent parts. A sequence of 12 operations must be carried out to complete each of the three parts. Tables 4.2, 4.3, 4.4 provide a detailed description of this RMS.

4.6.2 System Modeling Using Gen-RONs

Gen-RONs formalism specifies each RMS configuration as an L-P/T net. The initial population is generated randomly as a set of L-P/T nets. The proposed algorithm (Algorithm 1) iteratively applies selection, mutation, crossover, and evaluation, to identify optimal configurations. Using the previously described modeling technique, the abstract template model and the machine models are illustrated in Figures 4.17, 4.18, 4.20, and 4.19. Each configuration has a specific structure; however, a common sub-structure exists among all possible configurations (i.e., L-P/T nets of population generations). This common sub-structure serves as an abstract template for the RMS, specialized for each configuration. This template is illustrated in Figure 4.17. It shows the 12 operations, their order, the required commands (represented by double blue circles) and the machine configurations (represented by double red circles) for each operation.

Table 4.2: Matrix of the operations precedence relationship of the product three parts and Tools required for each part to be manufactured.

Part1	***OP1***	***OP2***	***OP3***			***Tools required***
OP1	0	0	0			1
OP2	0	0	1			2
OP3	1	0	0			1
Part2	***OP1***	***OP2***	***OP3***	***OP4***	***OP5***	***Tools required***
OP1	0	0	0	0	0	3
OP2	1	0	0	0	0	5
OP3	1	0	0	0	0	4
OP4	1	0	1	0	0	1
OP5	1	0	1	0	0	1
Part3	***OP1***	***OP2***	***OP3***	***OP4***		***Tools required***
OP1	0	0	0	0		3
OP2	1	0	0	0		4
OP3	1	0	0	0		4
OP4	1	1	1	0		2

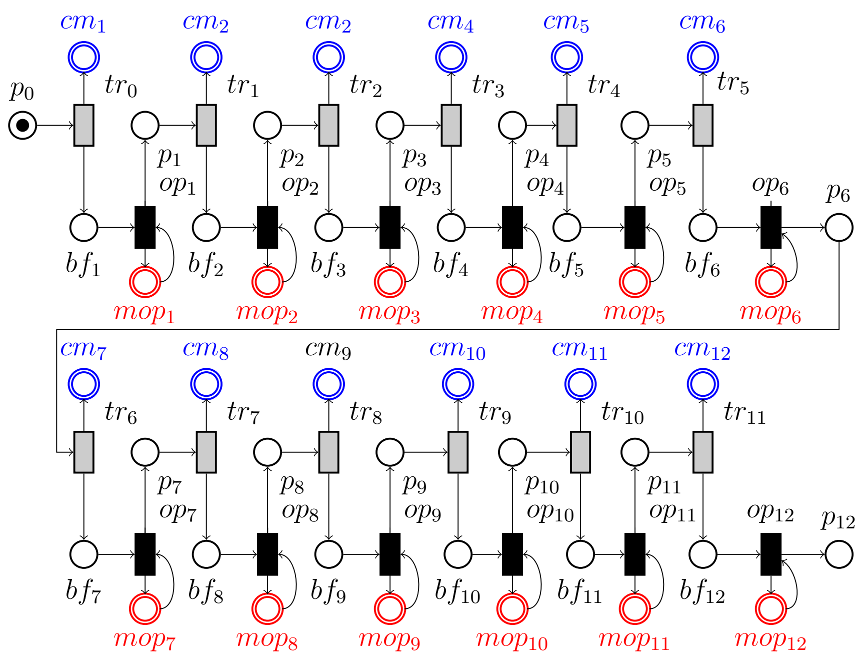


Figure 4.17: An abstract L-P/T net of an initial configuration of the RMS.

All operations must be performed using only three reconfigurable machines. These ma-

Table 4.3: Matrix of TADs required for each operation.

Part1	X	$-X$	Y	$-Y$	Z	$-Z$
OP1	1	1	0	0	0	0
OP2	0	0	0	0	1	1
OP3	1	0	0	0	0	0
Part2						
OP1	0	1	0	0	0	0
OP2	0	0	0	0	1	1
OP3	0	1	0	0	0	1
OP4	0	0	0	0	1	0
OP5	0	0	1	1	0	0
Part3						
OP1	0	0	0	0	0	
OP2	1	0	0	0	0	
OP3	1	0	0	0	0	
OP4	1	1	1	0	0	

Table 4.4: Matrix of configurations TADs and Tools available for each machine.

	X	$-X$	Y	$-Y$	Z	$-Z$	***Tools***
M_1							1, 4, 2
C1	1	1	1	0	0	0	
C2	1	0	1	0	1	0	
C3	1	1	0	1	0	1	
M_2							2, 3, 1, 6
C1	1	0	0	0	1	1	
C2	0	1	1	1	0	0	
C3	0	1	0	1	0	1	
C4	0	0	1	1	1	1	
M_3							1, 7, 5
C1	0	0	0	0	1	1	
C2	1	1	0	0	0	0	

chines are modeled by three L-P/T nets, as shown in Figures 4.18, 4.20, and 4.19.

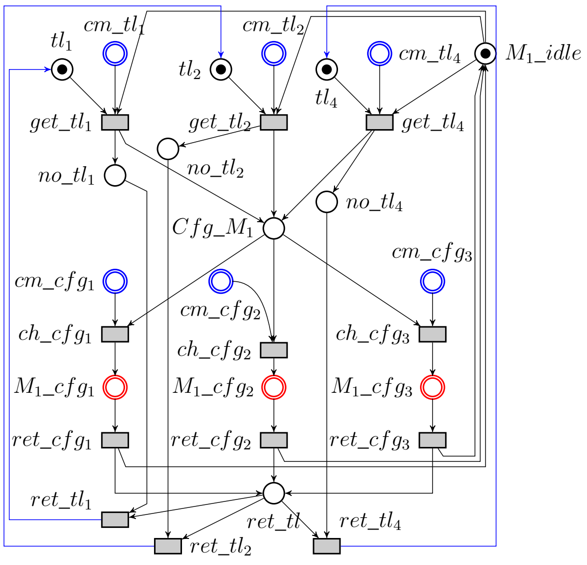


Figure 4.18: The L-P/T net of machine M_1.

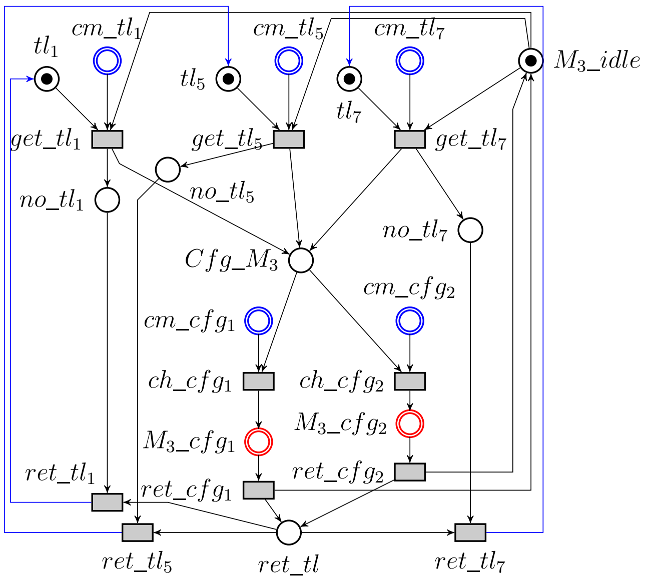


Figure 4.19: The L-P/T net of machine M_3.

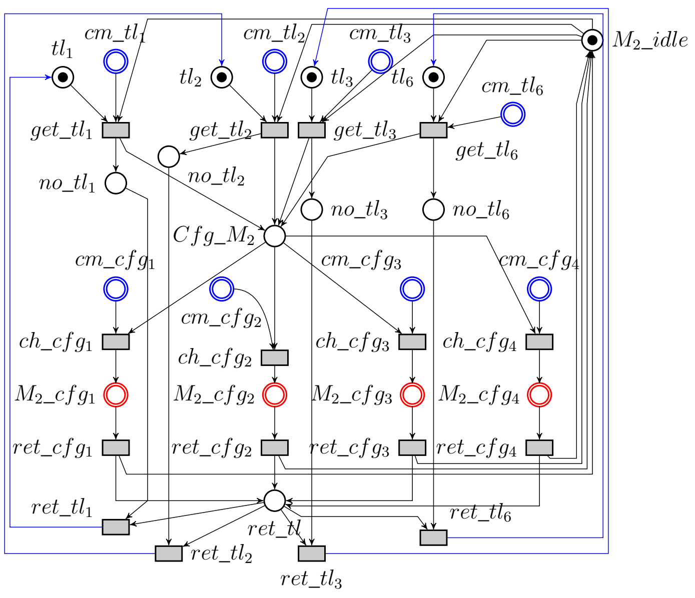


Figure 4.20: The L-P/T net of machine M_2.

4.6.3 Example of a Configuration L-P/T Net

Based on the description provided in section 4.6.1 and the set of P/T nets models, several configurations are possible. For example, configuration $Conf_0$ of the system may include: $M2$(C1) (performing OP2 in Part1), $M1$(C1) (performing OP3 and OP1 in Part1), $M3$(C1) (performing OP2 in Part2), $M1$(C3) (performing OP5 in Part2), M1(C2) (performing OP3 and OP4 in Part2), $M2$(C2) (performing OP1 in Part2), $M1$(C1) (performing OP4, OP2 and OP3 in Part3), an finally $M2$(C1) (performing OP1 in Part3). This configuration requires establishing a set of connections between the four models (depicted in Figures 4.17, 4.18, 4.20 and 4.19), as detailed in Table 4.5. Other configurations can be derived by updating connections between these models.

4.6.4 Performance Evaluation

Once the modeling of the RMS is completed using Gen-RONs formalism, Algorithm 1 (presented in Section 4.5) is employed to identify an optimal configuration based on the desired objective functions. This subsection presents the performance evaluation of the proposed formalism, divided into three parts: 1) a comparison with classical NSGA-II using the same genetic operator parameters; 2) a comparison using Taguchi's method; and finally 3) a comparative study with related works. The implementation of the NSGA-II based on Gen-RONs was conducted on Microsoft Windows 7 Professional with 64 bits, featuring a 2.50 GHz Intel(R) i5-2520M processor and 8.00 GB RAM. The programming tools used include Python Interpreter 3.4.0 and PyCharm IDE 2016.3.1.

4.6.4.1 Comparison with the Classical NSGA-II Using Identical Genetic Operator Parameters

In this subsection, a comparison is made between the computational time of the proposed approach and that of NSGA-II, used in [218]. An approach for encoding and decoding the individuals (solutions) for the optimization process using NSGA-II is proposed in [218]. The RMS case study, described previously, is adopted in [218] for consistency. A total of 25 runs were executed, featuring a population size of 40, a maximum of 15 generations, a crossover rate of 0.6 and a mutation rate of 0.08. Notably, the computational time for the NSGA-II method exceeds 3 hours (approximately 3 hours and 13 minutes). In clear contrast, the proposed approach demonstrates significantly enhanced performance, requiring only 26 minutes (about one minute per run). This significant difference in computational time can be primarily attributed to the complexity of the encoding and decoding procedures used in [218], where the decoding process is particularly CPU-intensive. The true Pareto fronts are depicted in Figure 4.21.

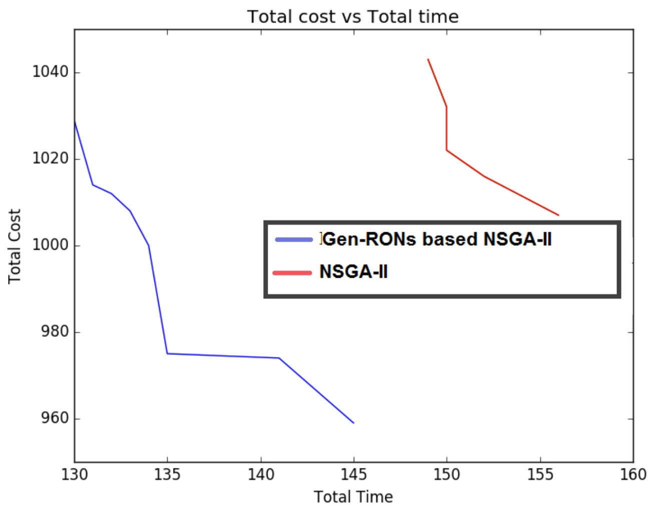


Figure 4.21: True Pareto in "Gen-RONs based NSGA-II" vs. NSGA-II.

4.6.4.2 Comparison Using Taguchi's Method

In this subsection, Taguchi's method is employed to determine the optimal solution after tuning the parameters and to compare *Gen-RONs-based NSGA-II* with classical NSGA-II. For the implementation of Taguchi's method, an experiment is conducted with the following four parameters: (i) population size, (ii) number of generations to stop the algorithm, (iii) crossover rate, and (iv) mutation rate, each having three levels (i.e., three sets of values), as shown in Table 4.6. Therefore, Taguchi's orthogonal arrays design L_9 is utilized, as depicted in Table 4.7. To ensure accuracy, 15 runs were performed for each experiment. Tables 4.8 and 4.9 illustrate the results of the fifteen runs of the proposed approach for the first and last experiments. The first experiment uses the parameter values of level 1: population size = 50, generation number = 25, crossover rate = 0.7 and mutation rate = 0.2. The last experiment uses the parameter values of: level 3 for population size (100) and generation number (50), level 2 for the crossover rate (0.8), and level 1 for the mutation rate (0.2). Similarly, the same experiments were conducted for classical NSGA-II. Tables 4.8 and 4.9 present the results.

Table 4.6: Level values of the studied parameters.

Parameters	Level 1	Level 2	Level 3
Population size	50	80	100
Generation number	25	30	50
Crossover rate	0.7	0.8	0.9
Mutation rate	0.2	0.08	0.01

Table 4.7: Taguchi's orthogonal arrays L_9.

Experiment	Pop_{size}	Nbr_{gen}	$Cros_{rate}$	Mut_{rate}
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

Table 4.8: NSGAII$_{Gen-RONs}$ and NSGA-II results of experiment 1.

Run	NSGA-II$_{GEn-RONs}$			NSGA-II		
	Cost average	Time average	solutions number	Cost average	Time average	Solutions number
1	760.66	149.33	3	1011.91	151.83	12
2	914.79	148.28	63	988	147	1
3	791.68	150.75	16	990.33	150	3
4	762.30	154.15	13	997.16	151.33	6
5	846.8	151.8	5	1016	145.5	2
6	954.75	154.0	32	1017	151	5
7	821.08	151.02	46	1009	149.8	5
8	777.0	151.7	10	1018.72	152	11
9	832.85	151.28	7	1011.66	148.33	3
10	816.25	155.5	4	1004.5	148	2
11	866.1	151.0	10	1009	149.83	6
12	765.57	151.57	7	997	148.16	6
13	784.33	150.83	6	985	146	1
14	929.72	150.36	11	993	149	1
15	828.27	152.09	11	991	150.66	6

Table 4.9: NSGAII$_{Gen-RONs}$ and NSGA-II results of expirement 9.

Run	NSGA-II$_{GEn-RONs}$			NSGA-II		
	Cost average	Time average	solutions number	Cost average	Time average	Solutions number
1	872.09	147.88	137	992	147	1
2	747.40	148.42	177	996.4	147.4	5
3	845.83	147.47	160	985	145	3
4	727.6	146.7	200	994.9	149.5	10
5	730.61	148.77	145	986	145	1
6	743.27	150.49	151	1019	146.85	14
7	767.4	148.95	40	989	146.6	5
8	743.20	148.39	175	997.5	145	2
9	732.49	148.01	184	1004	148.4	10
10	746.59	147.49	195	1005.09	149	11
11	748.58	147.17	17	992.4	149.2	5
12	741.12	147.67	195	998.81	148.72	11
13	746.70	146.53	179	989.25	146	4
14	877.62	148.35	139	1012.36	149.09	11
15	727.98	146.50	185	997.33	146.66	3

As seen in Tables 4.8 and 4.9, the number of solutions obtained by Gen-RONs-based NSGA-II exceeds that of classical NSGA-II. Furthermore, the best solutions in terms of cost and time are achieved by Gen-RONs. Specifically, average costs of 760.66 (in the first run of the first experiment) and 727.6 (in the fourth run of the last experiment) are recorded, with average times of 149.33 and 146.7, respectively. In contrast, classical NSGA-II shows average costs of 993 (in the fourteenth run of the first experiment) and 989 (in the seventh run of the last experiment). Although the average times for classical NSGA-II runs are similar to those of Gen-RONs, the costs are significantly higher.

The optimal Pareto fronts (i.e., the solutions sets of the last generation, number 50) from the final run of the experiment 9 for both algorithms, Gen-RONs-based NSGA-II and NSGA-II, are depicted in Figure 4.22.

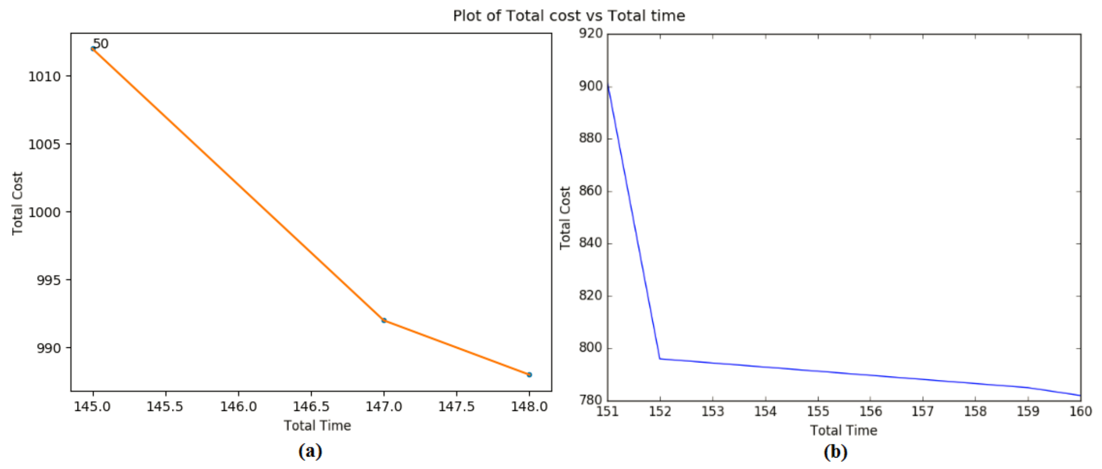


Figure 4.22: The optimal Pareto fronts (generation number 50) for (a) NSGA-II and (b) Gen-RONs based NSGA-II.

Using the performance metrics defined in Section 4.3.2.3, extensive experiments are conducted to tune the parameter of the genetic algorithm. The S/N ratio, a metric for assessing product performance sensitivity to noise [273], is computed. In this context, Taguchi's method is employed to identify the maximum S/N ratio, which is known as the bigger-the-best metric, as defined in Equation 4.15.

$$S/N = -10 \log \frac{1}{n} \sum_{i=1}^{n} \frac{1}{sum_i^2} \tag{4.15}$$

Where sum_i represents the summation of the two computed metrics (spacing and coverage) for each run. For example, Table 4.10 presents the metric values obtained in the first run for each experiment, long with their summation for both algorithms: Gen-RONs-based NSGA-II and classical NSGA-II. Tables 4.11 and 4.12 show the S/N ratios for the two algorithms.

Table 4.10: Gen-RONs based NSGA-II metrics results of the first run.

Exp		NSGA-II$_{Gen-RONs}$			NSGA-II	
	SP metric	CM metric	Sum	SP metric	CM metric	Sum
1	1.2079	1	2.2079	0.6088	0	0.6088
2	0.8971	1	1.8971	0.7460	0	0.7460
3	1.9591	1	2.9591	0.6768	1	1.6768
4	1.9746	0.3157	2.2904	0.6251	1	1.6251
5	1.9746	1	2.9746	0.7997	0.1428	0.9425
6	1.9746	1	2.9746	0.8061	0.125	0.9311
7	1.2576	1	2.2576	0.7326	0.6666	1.3993
8	1.9797	1	2.9797	0.6902	0.1666	0.8569
9	1.9797	1	2.9797	0.8551	0	0.8551

From Tables 4.11 and 4.12, the S/N ratio values for each parameter by level can be deduced for both algorithms: Gen-RONs-based NSGA-II and NSGA-II. These values are detailed in Tables 4.13 and 4.14, respectively.

Table 4.13: Gen-RONs based NSGA-II S/N ratio values for parameters by levels.

Level	Population size	Generation number	Crossover rate	Mutation rate
1	7.05	6.42	7.28	7.52
2	7.43	6.88	7.31	7.34
3	7.47	8.65	7.35	7.09

Table 4.14: NSGA-II S/N ratio values for parameters by levels.

Level	Population size	Generation number	Crossover rate	Mutation rate
1	-0.11	-0.23	-0.55	0,01
2	-1.03	-0.42	0.05	-1.09
3	0.42	-0.06	0.08	0.34

Figures 4.23 and 4.24 illustrate the effect of the parameters on both algorithms.

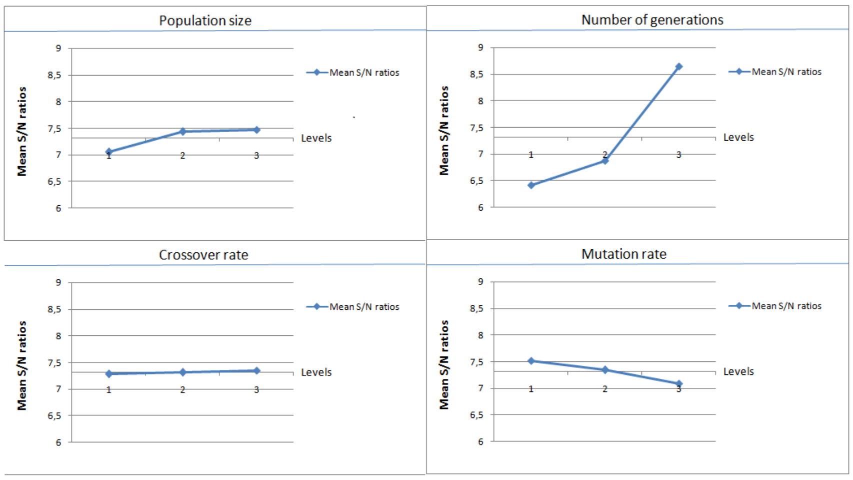


Figure 4.23: The mean of S/N ratio plots for each level of Gen-RONs based NSGA-II parameters.

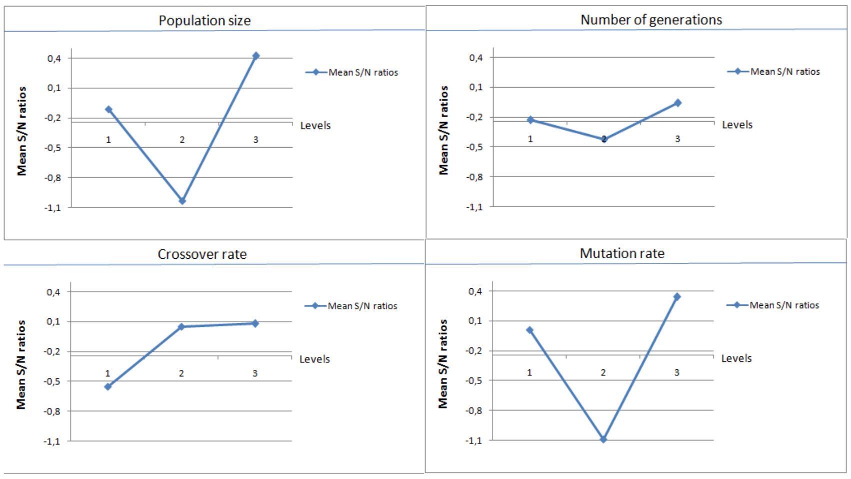


Figure 4.24: The mean of S/N ratio plots for each level of NSGA-II parameters.

The effect of parameters on Gen-RONs-based NSGA-II algorithm is shown in Figure 4.23. The analysis reveals that the highest mean S/N ratio value for the population size parameter occurs at level 3, and the same level applies to number of generations. In contrast, the S/N ratio values for the crossover rate and mutation rate are at level 2 and 1, respectively. Figure 4.23 indicates that the generations number parameter yields the maximum S/N ratio; thus, generation number of 50 is selected. This choice is expected to enhance the algorithm's effectiveness, given that the other parameters, while less impactful on the S/N ratio, significantly influence the overall performance of the algorithm.

For the NSGA-II algorithm, the effect of parameters is illustrated in Figure 4.24. The analysis shows that the population size and mutation rate parameters yield the highest S/N ratios. Consequently, a population size of 100 and a mutation rate of 0.2 are selected. These values are expected to enhance the algorithm's effectiveness, as the other parameters, while having minimal impact on the S/N ratio, significantly influence the overall performance of the algorithm.

After computing the S/N ratio, the highest S/N value corresponds to the best combination of the parameter values. Based on Tables 4.11 and 4.12, the largest S/N ratio obtained from both algorithms occurs Experiment 9. Therefore, the optimal parameters are the underlined values in Table 4.15.

Table 4.15: Tuned parameters for both algorithms.

parameters	Level 1	Level 2	Level 3
Population size	50	80	**100**
Generation number	25	30	**50**
Crossover rate	0.7	**0.8**	0.9
Mutation rate	**0.2**	0.08	0.01

4.6.4.3 Comparative qualitative study

Finally, Table 4.16 summarizes some characteristics and qualitative differences between this work and the most closely related studies.

Table 4.5: Required connections to model $Conf_0$

	Connected	***communication***	***places***	
Operation	Abstract model	M_1 model	M_2 model	M_3 model
(OP2: Part1) on	cm_1	_	cm_cfg_1,	_
M_2(C1)			cm_tl_2	
	mop_1	_	$M_2\_cfg_1$	_
(OP3: Part1) on	cm_2	cm_cfg_1,	_	_
M_1(C1)		cm_tl_1		
	mop_2	$M_1\_cfg_1$	_	_
(OP1: Part1) on	cm_3	cm_cfg_1,	_	_
M_1(C1)		cm_tl_2		
	mop_3	$M_1\_cfg_1$	_	_
(OP2: Part2) on	cm_4	_	_	cm_cfg_1,
M_3(C1)				cm_tl_5
	mop_4	_	_	$M_3\_cfg_1$
(OP5: Part2) on	cm_5	_	_	cm_cfg_3,
M_1(C3)				cm_tl_1
	mop_5	_	_	$M_3\_cfg_3$
(OP3: Part2) on	cm_6	cm_cfg_2,	_	_
M_1(C2)		cm_tl_4		
	mop_6	$M_1\_cfg_2$	_	_
(OP4: Part2)	cm_7	cm_cfg_2,	_	
		M_1(C2)	cm_tl_1	
	mop_7	$M_1\_cfg_2$	_	_
(OP1: Part2) on	cm_8	_	cm_cfg_2,	_
M_2(C2)			cm_tl_3	
	mop_8	_	$M_2\_cfg_2$	_
(OP4: Part3) on	cm_9	cm_cfg_1,	_	_
M_1(C1)		cm_tl_2		
	mop_9	$M_1\_cfg_1$	_	_
(OP2: Part3) on	cm_10	cm_cfg_1,	_	_
M_1(C1)		cm_tl_4		
	mop_10	$M_1\_cfg_1$	_	_
(OP3: Part3) on	cm_11	cm_cfg_1,	_	_
M_1(C1)		cm_tl_4		
	mop_11	$M_1\_cfg_1$	_	_
(OP1: Part3) on	cm_12	_	cm_cfg_1,	_
M_2(C1)			cm_tl_3	
	mop_12	_	$M_2\_cfg_1$	_

Table 4.11: Experiments result of the Gen-RONs based NSGA-II.

Metrics sum	Exp1	Exp2	Exp3	Exp4	Exp5	Exp6	Exp7	Exp8	Exp9
sum1	2.207923	1.897102	2.959183	2.290473	2.9746831	2.974683	2.257651	2.979797	2.97979
sum2	2.795918	2.126545	1.959183	1.911267	2.309988	2.974683	2.076675	2.247128	2.979797
sum3	2.041924	2.261522	2.959183	1.989359	2.949367	2.974683	1.955162	2.011660	2.979797
sum4	1.988754	1.908263	2.959183	2.122383	2.565352	2.974683	2.159906	1.850763	2.979797
sum5	2.007621	2.123203	2.959183	2.159144	2.062122	2.415267	2.211966	2.181216	2.979797
sum6	1.738962	2.201417	1.969148	2.010019	2.370845	2.974683	2.959595	2.205863	2.939393
sum7	2.527298	2.918367	2.448765	2.019649	2.298192	2.974683	2.125042	2.041848	2.220573
sum8	2.213756	2.918367	2.959183	1.848057	2.325029	2.004360	2.155193	2.298596	2.979797
sum9	2.106817	1.981631	2.375293	2.140329	1.934840	2.974683	2.281718	1.970958	2.979797
sum10	2.019110	1.927755	2.959183	1.921316	2.370909	2.974683	2.050418	2.979797	2.979797
sum11	2.202680	2.170674	2.959183	1.922367	2.148975	2.148975	2.318047	2.318047	2.147233
sum12	2.049843	1.988135	2.405881	2.062407	2.166575	2.974683	2.057653	1.958387	2.979797
sum13	2.035595	2.273932	2.184990	2.323042	2.113503	2.974683	1.910912	2.335480	2.979797
sum14	1.981212	2.118557	2.959183	1.937694	2.300580	2.974683	2.005897	2.183397	2.979797
sum15	1.892188	2.141895	2.959183	2.898734	2.297459	2.900214	2.125219	2.028874	2.979797
S/N ratio	6.329805	6.635045	8.195981	6.314578	7.240627	8.753045	6.634964	6.781311	9.003407

Table 4.12: Experiments result of the NSGA-II.

Metrics sum	Exp1	Exp2	Exp3	Exp4	Exp5	Exp6	Exp7	Exp8	Exp9
sum1	0.608838	0.746082	1.676841	1.625173	0.942517	0.931123	1.399335	0.856915	0.855100
sum2	0.630358	0.653811	1.586067	0.939181	0.841215	1.294556	0.768235	1.716581	1.318659
sum3	1.043806	1.784253	0.902421	1.081761	1.064970	1.721852	1.423027	0.642098	1.490878
sum4	0.805102	1.373085	0.998981	1.405153	1.109585	1.235853	0.695442	0.707525	0.870646
sum5	1.767910	0.973783	0.864435	1.157427	0.736598	0.740122	1.047722	1.688668	1.764139
sum6	1.247485	0.649334	1.207050	1.805257	0.756605	0.823010	1.790439	1.640093	1.615704
sum7	1.044450	1.542687	1.180931	1.100750	1.496083	1.144016	1.208074	1.608076	1.255215
sum8	0.536280	1.722835	0.723407	1.058635	0.710047	1.040294	0.671102	0.652262	1.894098
sum9	1.652340	0.993969	1.811730	1.658635	1.213786	1.647117	1.656357	1.551326	1.683055
sum10	1.574681	0.621739	1.070918	0.657518	0.756222	0.889321	0.904397	1.047455	1.595377
sum11	0.945542	1.379780	1.684810	1.681605	1.133535	0.874279	1.718248	1.750369	0.790294
sum12	1.724419	0.698009	1.360885	0.607723	1.315060	0.287140	0.908594	1.651332	1.589821
sum13	1.776935	1.635992	1.492963	0.684289	1.484753	1.060523	1.255773	1.726219	1.045506
sum14	1.713393	0.675918	0.873695	1.252406	2.004485	0.764409	1.797188	0.698319	0.773811
sum15	1.087458	1.681260	1.402053	0.864450	0.617654	1.772514	0.656395	1.760779	1.108000
S/N ratio	-0.481465	-0.853340	0.979274	-0.140561	-0.634741	-2.341944	-0.085110	0.203664	1.164607

Table 4.16: Comparative qualitative study.

Work	Approach	Use of PNs	PNs class	Systems	Optimization	Metrics	Taguchi
[249]	GA+PNs	Simulation	Simplified Timed Petri Nets	FMS	Single-objective	Scheduling time	Yes
[250]	GA+PNs	PNs model GAs	Colored Petri Nets	Business Process	Multi-objective	Scheduling time, total cost	No
[225]	GA	Not used	Not used	Mixed-model two-sided assembly line	Multi-objective	Line efficiency, smoothness index, cost	No
[222]	GA	Not used	Not used	RMS	Multi-objective	Tardiness, cost	No
[267]	GA	Not used	Not used	Vendor managed inventory model	Multi-objective	Inventory cost, warehouse space	Yes
[232]	PNs+linear programming	PNs model production process	First-Order Hybrid Petri nets	Low-Automated production Process	Single-objective	Process speeds	No
[252]	GA+PNs	PNs model scheduling	Place/Transition Petri nets	Flexible job shop	Single-objective	Makespan	No
Gen-RONs	GA+PNs	PNs model chromosomes	Reconfigurable Object Nets	RMS	Multi-objective	Cost, completion time	Yes

"PNs model GAs" means that Petri nets are used to model the entire genetic algorithms.
"PNs model scheduling" means that Petri nets are used to model the scheduling problem.
"PNs model chromosomes" means that Petri nets are used to model only the chromosomes (i.e., RMS configurations).

4.6.5 Discussion

This subsection presents a comprehensive discussion of the proposed approach, focusing on its scalability, impact, and connections to related work.

4.6.5.1 Scalability of the Proposed Approach

Addressing scalability is crucial for ensuring the RMSs' adaptability to volatile market demands and uncertain future requirements. The work reported in [274] focuses on the importance of evaluating the scalability of RMSs during the design phase. It integrates modular design principles in methodologies for scalability assessment. By incorporating formal methods, the work reported in [275] uses discrete timed Petri nets (DTPNs) to model and evaluate RMSs' scalability, resilience and robustness. It provides a comprehensive method for assessing the system adaptability. In this subsection of the present work, the authors emphasize modularity and the use of formal methods (Petri nets), as further validated individually in the aforementioned studies, to discuss the scalability.

Increasing the number of operations in an RMS can increase the number of transitions in the RMS Petri net model, thus, potential increasing in the number of places and transitions that model the machines behaviors too. The user must consider the RMS model as sub-models. Each module (i.e., sub-Petri net model) represents the operation to be performed and the machine (with its configurations if needed). Then, the entire RMS model is constructed by connecting these modules using gluing places. Thus, the scalability of the proposed approach touches three phases. (i) Modeling phase: This modularity reduces the effort and time of modeling. (ii) Analysis phase: the modularity of the RMS Petri net model provides a rational analysis. The TINA tool (a symbolic model-checker) optimizes the checking phase. (iii) Optimization phase: the impact of the number of operation influences only on the length of a chromosome. The proposal uses a pre-existing genetic algorithm. In the present work, the authors choose NSGA-II [136] as a genetic algorithm. Since this algorithm is considered as a meta-heuristic with a complexity of polynomial time computational, it is applicable for mid-to-large-scale problems. Thus, Gen-RONs formalism, in this case, is applicable for mid-to-large-scale problems.

4.6.5.2 Discussion of the impact of the proposed approach

The minimal impact of evolutionary operations on algorithm performance in the current proposed method is primarily due to strict constraints imposed by reconfigurability requirements ensuring system properties such as deadlock-freeness and liveness are maintained. The current proposed approach balances these constraints by employing carefully designed genetic operators suited for reconfigurability challenges. While improvements in performance may seem incremental, they are significant in preserving essential system properties and achieving feasible reconfigurations. Further work could involve developing more detailed metrics that reflect the dual objectives of performance optimization and property preservation.

4.6.5.3 Connection with related work

Advancing the field of RMSs requires both cutting-edge optimization techniques and strategic implementation frameworks. The novel formalism Gen-RONs combines PNs and GAs to assess formally RMS and optimize its configurations dynamically. This ensures not only fluctuating production demands but also preserves essential system properties such as deadlock-freeness. Complementing the current proposed optimization-centric formal approach, the research presented in [182] offers a strategic perspective by identifying and prioritizing key RMS enablers (RMSEs). Through a detailed framework, the study provides the practical implementation of RMSs focusing on critical factors necessary for successful adoption and operation. The synergy between this study and the current proposed approach lies in the integration of these two approaches. While Gen-RONs focus on computational efficiency and real-time adaptability, the strategic guidance from the study [182] provides a robust foundation for practical implementation. This combination bridges the gap between theoretical optimization and real-world application ensuring that the benefits of advanced RMS configurations are fully realized in practical settings. By leveraging both computational and strategic strengths, this integration significantly enhances the performance and adoption of RMSs offering a comprehensive solution to the challenges faced by modern manufacturing environments.

4.7 Conclusion

The design and implementation of reconfigurable manufacturing systems (RMSs) present new challenges for both industry and researchers, including issues related to concurrency, synchronization, distribution, optimization, and maintenance. To ensure the reliability of RMSs and achieve optimal performance, researchers typically employ Petri nets [193] for formal modeling alongside evolutionary and genetic methods for optimization. Formal methods are frequently used for the formal specification of RMSs, analysis of their qualitative properties (deadlock-freedom, liveness and safety), and evaluation of their quantitative metrics (including throughput, time, and cost). Evolutionary and genetic methods have been explored to address optimization problems in RMSs, specifically to ensure that the reconfiguration of an RMS meets the required objectives. Traditionally, evolutionary/genetic approaches and Petri nets based methods have been applied independently. However, few studies have attempted to unify these two approaches.

In the present work, the authors have combined high level Petri nets with genetic algorithm. The newly proposed approach is called Gen-RONs (genetic reconfigurable object nets), which extends the previous formalism known as RONs (reconfigurable object nets) [198]. The objective is to integrate the basic operators of genetic algorithms, mutation and crossover, into the Petri net formalism. Consequently, formal definitions of these two operators are established within the Petri nets framework, treating both mutation and crossover as graph transformation operators defined over Place/Transition (P/T) nets. Building on these new formalizations, a genetic algorithm, Gen-RONs-based NSGA-II, is proposed. In this algorithm, chromosomes are represented as labeled P/T nets. This formalism and algorithm are utilized to optimize reconfigurable manufacturing systems, and the results obtained underscore the significance of the proposed approach in comparison to existing methods.

This study opens several promising perspectives at both theoretical and practical levels. Theoretically, future research will focus on the preservation of various critical properties within reconfigurable manufacturing systems. These systems are characterized by key quantitative properties (such as time, cost, and throughput) that can be maintained during configuration changes. Currently no formalism exists to ensure the preservation of these essential properties, presenting a significant challenge for the field. Extending reconfigurable Petri nets to address this issue represents a vital perspective for future investigation. On the practical side, the authors are dedicated to enhancing the tools developed for optimizing systems through the Gen-RONs approach, thereby improving the efficiency and effectiveness of RMS implementations.

GENERAL CONCLUSION

Thesis summary

In an era of increasing reliance on complex systems, ensuring their correctness and reliability is no longer optional, it is essential to prevent catastrophic failures that could disrupt industries or endanger lives. Reconfigurable systems, designed to adapt dynamically to evolving requirements, present both immense opportunities and critical challenges. For example, in industrial settings, undetected flaws in reconfigurable manufacturing systems could lead to production defects, equipment failures, or even workplace hazards. Verification of such systems is vital to safeguarding their operation and ensuring that adaptability does not compromise safety or efficiency. This thesis responds to these pressing needs by providing rigorous methodologies to verify, optimize, and enhance reconfigurable systems.

The research contributes significantly to the field, integrating formal methods and advanced optimization strategies to address the challenges of system design, verification, and optimization. By exploring both theoretical foundations and practical implementations, it lays the groundwork for systems that are not only adaptable but also robust, efficient, and safe. These innovations have the potential to revolutionize domains such as manufacturing, IoT, and networking, where reconfigurability is a critical enabler of innovation and resilience.

Through targeted case studies, the thesis demonstrates the transformative impact of combining formal verification techniques with optimization algorithms. It provides practical frameworks for addressing complex, real-world challenges, ensuring that reconfigurable systems meet their performance and safety requirements even under dynamic conditions.

This thesis makes the following key contributions:

- ***Advanced Modeling Framework for Mobile WSNs:*** Development of a robust framework employing UPPAAL SMC and probabilistic automata to model dynamic behaviors and ensure reliable communication protocols.
- ***Comprehensive Verification Techniques for IoT Protocols:*** Introduction of PCTL-based methodologies to verify the correctness of IoT protocols like MQTT, significantly enhancing their trustworthiness.
- ***Enhanced Optimization Strategies with Properties Preservation in Gen-RONs:*** Integration of Gen-RONs into the NSGA-II genetic algorithm, ensuring optimization while preserving essential system properties during crossover and mutation.
- ***Formal Modeling Using Reconfigurable Object Nets:*** Establishment of a structured framework for modeling and verifying reconfigurable manufacturing systems, addressing complexities in dynamic environments with a focus on analysis and optimization.

Open Issues and Future Research Directions

Despite these contributions, several open issues and future research directions remain:

- Scalability: Future research should focus on scaling our modeling techniques to handle larger and more complex systems, ensuring applicability as system requirements grow.
- Integration with Machine Learning: Exploring the incorporation of machine learning techniques into formal methods could enhance predictive capabilities and further optimize system performance.

- Real-Time Adaptation: Investigating real-time verification methods that allow systems to dynamically adapt to changing conditions will be crucial, particularly in fast-evolving environments like IoT.
- Cross-Domain Applications: Expanding the applicability of our methodologies to other fields, such as autonomous systems and smart cities, could provide insights that enhance the utility of formal methods across diverse industries.

Bibliography

[1] Tayfur Altiok. *Performance analysis of manufacturing systems*. Springer Science & Business Media, 1997.

[2] Daniel D Gajski, Frank Vahid, Sanjiv Narayan, and Jie Gong. *Specification and design of embedded systems*. Prentice-Hall, Inc., 1994.

[3] Gabriel Fornari and Valdivino Alexandre de Santiago Júnior. Dynamically reconfigurable systems: a systematic literature review. *Journal of Intelligent & Robotic Systems*, 95:829–849, 2019.

[4] Scott Ferguson, Afreen Siddiqi, Kemper Lewis, and Olivier L de Weck. Flexible and reconfigurable systems: Nomenclature and review. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 48078, pages 249–263, 2007.

[5] CH Koo, M Vorderer, S Junker, S Schröck, and A Verl. Challenges and requirements for the safety compliant operation of reconfigurable manufacturing systems. *Procedia CIRP*, 72:1100–1105, 2018.

[6] P-Y Piriou, J-M Faure, and J-J Lesage. From safety analysis of reconfigurable systems to design of fault-tolerant control strategies. In *2016 3rd Conference on Control and Fault-Tolerant Systems (SysTol)*, pages 624–629. IEEE, 2016.

[7] Jim Woodcock, Peter Gorm Larsen, Juan Bicarregui, and John Fitzgerald. Formal methods: Practice and experience. *ACM computing surveys (CSUR)*, 41(4):1–36, 2009.

[8] Edmund M Clarke. Model checking. In *Foundations of Software Technology and Theoretical Computer Science: 17th Conference Kharagpur, India, December 18–20, 1997 Proceedings 17*, pages 54–56. Springer, 1997.

[9] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[10] Lothar M Schmitt. Theory of genetic algorithms ii: models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary fitness function under scaling. *Theoretical Computer Science*, 310(1-3):181–231, 2004.

[11] Irene Ring. Evolutionary strategies in environmental policy. *Ecological Economics*, 23(3):237–249, 1997.

[12] Hisashi Tamaki, Hajime Kita, and Shigenobu Kobayashi. Multi-objective optimization by genetic algorithms: A review. In *Proceedings of IEEE international conference on evolutionary computation*, pages 517–522. IEEE, 1996.

[13] James C Lyke, Christos G Christodoulou, G Alonzo Vera, and Arthur H Edwards. An introduction to reconfigurable systems. *Proceedings of the IEEE*, 103(3):291–317, 2015.

[14] Amro M Farid. Measures of reconfigurability and its key characteristics in intelligent manufacturing systems. *Journal of intelligent manufacturing*, 28(2):353–369, 2017.

[15] Durga Prasad and SC Jayswal. A review on flexibility and reconfigurability in manufacturing system. *Innovation in Materials Science and Engineering: Proceedings of ICEMIT 2017, Volume 2*, pages 187–200, 2018.

[16] Murillo Augusto da Silva Ferreira, Guilherme Cano Lopes, Esther Luna Colombini, and Alexandre da Silva Simões. A novel architecture for multipurpose reconfigurable unmanned aerial vehicle (uav): concept, design and prototype manufacturing. In *2018 Latin American robotic symposium, 2018 Brazilian symposium on robotics (SBR) and 2018 workshop on robotics in education (WRE)*, pages 443–450. IEEE, 2018.

[17] Raji George, CRS Kumar, SA Gangal, and Makarand Joshi. A low profile reconfigurable antenna for defense aircrafts. *SN Applied Sciences*, 1(6):608, 2019.

[18] Safa Guellouz, Adel Benzina, Mohamed Khalgui, Georg Frey, Zhiwu Li, and Valeriy Vyatkin. Designing efficient reconfigurable control systems using iec61499 and symbolic model checking. *IEEE Transactions on Automation Science and Engineering*, 16(3):1110–1124, 2018.

[19] Aneesh Paul, Rohan Chauhan, Rituraj Srivastava, and Mriganka Baruah. Advanced driver assistance systems. Technical report, SAE Technical Paper, 2016.

[20] Yoram Koren, Uwe Heisel, Francesco Jovane, Toshimichi Moriwaki, Gumter Pritschow, Galip Ulsoy, and Hendrik Van Brussel. Reconfigurable manufacturing systems. *CIRP Annals-Manufacturing Technology*, 48(2):527–540, 1999.

[21] Normaziah A Aziz, Teddy Mantoro, M Aiman Khairudin, et al. Software defined networking (sdn) and its security issues. In *2018 International conference on computing, engineering, and design (ICCED)*, pages 40–45. IEEE, 2018.

[22] Christian Tipantuna and Paúl Yanchapaxi. Network functions virtualization: An overview and open-source projects. In *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*, pages 1–6. IEEE, 2017.

[23] Tammara Massey, Foad Dabiri, Roozbeh Jafari, Hyduke Noshadi, Philip Brisk, William Kaiser, and Majid Sarrafzadeh. Towards reconfigurable embedded medical systems. In *2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPnP 2007)*, pages 178–180. IEEE, 2007.

[24] Jacek Feczko, Michal Manka, Pawel Krol, Mariusz Giergiel, Tadeusz Uhl, and Andrzej Pietrzyk. Review of the modular self reconfigurable robotic systems. In *2015 10th International Workshop on Robot Motion and Control (RoMoCo)*, pages 182–187. IEEE, 2015.

[25] Helen C Leligou, Luis Redondo, Theodore Zahariadis, Daniel Rodriguez Retamosa, Panagiotis Karkazis, Ioannis Papaefstathiou, and Stamatis Voliotis. Reconfiguration in

wireless sensor networks. In *2010 Developments in E-systems Engineering*, pages 59–63. IEEE, 2010.

[26] Javad Rezazadeh, Marjan Moradi, and Abdul Samad Ismail. Mobile wireless sensor networks overview. *International Journal of Computer Communications and Networks*, 2(1):17–22, 2012.

[27] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services & Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond.

[28] Khalida S Rijab and Saif Muqdad Sadiq. Implementing a reconfigurable internet of things nodes using non-ip network based on wireless sensor network. 2019.

[29] Ramaasamy Velmani. Mobile wireless sensor networks: an overview. *Wireless Sensor Networks. IntechOpen Limited: London, UK*, 2017.

[30] Ricardo Silva, Jorge Sa Silva, and Fernando Boavida. Mobility in wireless sensor networks–survey and proposal. *Computer Communications*, 52:1–20, 2014.

[31] Wei Ye, John Heidemann, and Deborah Estrin. An energy-e cient mac protocol for wireless sensor networks. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, page 1, 2009.

[32] Huan Pham and Sanjay Jha. Addressing mobility in wireless sensor media access protocol. *International Journal of Distributed Sensor Networks*, 1(2):269–280, 2005.

[33] Prasad Raviraj, Hamid Sharif, Michael Hempel, and Song Ci. Mobmac-an energy efficient and low latency mac for mobile wireless sensor networks. In *2005 Systems Communications (ICW'05, ICHSN'05, ICMCS'05, SENET'05)*, pages 370–375. IEEE, 2005.

[34] Sung-Chan Choi, Jang-Won Lee, and Yeonsoo Kim. An adaptive mobility-supporting mac protocol for mobile sensor networks. In *VTC Spring 2008-IEEE Vehicular Technology Conference*, pages 168–172. IEEE, 2008.

[35] SA Hameed, EM Shaaban, HM Faheem, and MS Ghoniemy. Mobility-aware mac protocol for delay-sensitive wireless sensor networks. In *2009 International Conference on Ultra Modern Telecommunications & Workshops*, pages 1–8. IEEE, 2009.

[36] Bilal Muhammad Khan and Falah H Ali. Collision free mobility adaptive (cfma) mac for wireless sensor networks. *Telecommunication Systems*, 52:2459–2474, 2013.

[37] Muneeb Ali, Tashfeen Suleman, and Zartash Afzal Uzmi. Mmac: A mobility-adaptive, collision-free mac protocol for wireless sensor networks. In *PCCC 2005. 24th IEEE International Performance, Computing, and Communications Conference, 2005.*, pages 401–407. IEEE, 2005.

[38] Amre El-Hoiydi and J-D Decotignie. Wisemac: an ultra low power mac protocol for the downlink of infrastructure wireless sensor networks. In *Proceedings. ISCC 2004. Ninth International Symposium on Computers And Communications (IEEE Cat. No. 04TH8769)*, volume 1, pages 244–251. IEEE, 2004.

[39] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, 2004.

[40] Michael Buettner, Gary V Yee, Eric Anderson, and Richard Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320, 2006.

[41] Yanjun Sun, Omer Gurewitz, and David B Johnson. Ri-mac: a receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 1–14, 2008.

[42] Prabal Dutta, Stephen Dawson-Haggerty, Yin Chen, Chieh-Jan Mike Liang, and Andreas Terzis. Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 1–14, 2010.

[43] Papa Dame Ba, Bamba Gueye, Ibrahima Niang, and Thomas Noel. Mox-mac: A low power and efficient access delay for mobile wireless sensor networks. In *2011 4th Joint IFIP Wireless and Mobile Networking Conference (WMNC 2011)*, pages 1–6. IEEE, 2011.

[44] Bing Liu, Ke Yu, Lin Zhang, and Huimin Zhang. Mac performance and improvement in mobile wireless sensor networks. In *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, volume 3, pages 109–114. IEEE, 2007.

[45] Luís Bernardo, H Agua, M Pereira, Rodolfo Oliveira, Rui Dinis, and Paulo Pinto. A mac protocol for mobile wireless sensor networks with bursty traffic. In *2010 IEEE Wireless Communication and Networking Conference*, pages 1–6. IEEE, 2010.

[46] MJ Handy, Marc Haase, and Dirk Timmermann. Low energy adaptive clustering hierarchy with deterministic cluster-head selection. In *4th international workshop on mobile and wireless communications network*, pages 368–372. IEEE, 2002.

[47] G Santhosh Kumar, Paul MV Vinu, and K Poulose Jacob. Mobility metric based leach-mobile protocol. In *2008 16th International conference on advanced computing and communications*, pages 248–253. IEEE, 2008.

[48] Petros Spachos, Dimitris Toumpakaris, and Dimitrios Hatzinakos. Angle-based dynamic routing scheme for source location privacy in wireless sensor networks. In *2014 IEEE 79th Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2014.

[49] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile computing*, pages 153–181. Springer, 1996.

[50] Muni Venkateswarlu Kumaramangalam, Kandasamy Adiyapatham, and Chandrasekaran Kandasamy. Zone-based routing protocol for wireless sensor networks. *International scholarly research notices*, 2014(1):798934, 2014.

[51] Ilias Leontiadis and Cecilia Mascolo. Geopps: Geographical opportunistic routing for vehicular networks. In *2007 IEEE international symposium on a world of wireless, mobile and multimedia networks*, pages 1–6. Ieee, 2007.

[52] Omair Fatmi and Deng Pan. Distributed multipath routing for data center networks based on stochastic traffic modeling. In *Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control*, pages 536–541. IEEE, 2014.

[53] Mohammad Mansour, Amal Gamal, Ahmed I Ahmed, Lobna A Said, Abdelmoniem Elbaz, Norbert Herencsar, and Ahmed Soltan. Internet of things: A comprehensive overview on protocols, architectures, technologies, simulation tools, and future directions. *Energies*, 16(8):3465, 2023.

[54] Riccardo Bonetto, Nicola Bui, Vishwas Lakkundi, Alexis Olivereau, Alexandru Serbanati, and Michele Rossi. Secure communication for smart iot objects: Protocol stacks, use cases and practical examples. In *2012 IEEE international symposium on a world of wireless, mobile and multimedia networks (WoWMoM)*, pages 1–7. IEEE, 2012.

[55] Keyur K Patel, Sunil M Patel, and P Scholar. Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges. *International journal of engineering science and computing*, 6(5), 2016.

[56] Mqtt version 3.1.1 plus errata 01. `http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html`. Accessed: 2017-July-09.

[57] Zach Shelby, Klaus Hartke, and Carsten Bormann. The constrained application protocol (coap). Technical report, 2014.

[58] Jeferson Rodrigues Cotrim and João Henrique Kleinschmidt. Lorawan mesh networks: A review and classification of multihop communication. *Sensors*, 20(15):4273, 2020.

[59] Farzad Samie, Lars Bauer, and Jörg Henkel. Iot technologies for embedded computing: A survey. In *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, pages 1–10, 2016.

[60] Shahin Farahani. *ZigBee wireless networks and transceivers*. newnes, 2011.

[61] Syed Rafiul Hussain, Shagufta Mehnaz, Shahriar Nirjon, and Elisa Bertino. Secure seamless bluetooth low energy connection migration for unmodified iot devices. *IEEE Transactions on Mobile Computing*, 17(4):927–944, 2017.

[62] Yoram Koren, Xi Gu, and Weihong Guo. Reconfigurable manufacturing systems: Principles, design, and future trends. *Frontiers of Mechanical Engineering*, 13:121–136, 2018.

[63] Zhuming M Bi, Sherman YT Lang, Weiming Shen, and Lihui Wang. Reconfigurable manufacturing systems: the state of the art. *International journal of production research*, 46(4):967–992, 2008.

[64] Abdelkrim R Yelles-Chaouche, Evgeny Gurevsky, Nadjib Brahimi, and Alexandre Dolgui. Reconfigurable manufacturing systems from an optimisation perspective: a focused review of literature. *International Journal of Production Research*, 59(21):6400–6418, 2021.

[65] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1):60–65, 2001.

[66] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[67] Kathrin Hoffmann, Hartmut Ehrig, and Till Mossakowski. High-level nets with nets and rules as tokens. In Gianfranco Ciardo and Philippe Darondeau, editors, *Proc. Applications and Theory of Petri Nets 2005*, pages 268–288, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[68] Farid Arfi, Anne-Lise Courbis, Thomas Lambolais, François Bughin, and Maurice Hayot. Formal verification of a telerehabilitation system through an abstraction and refinement approach using uppaal. *IET Software*, 17(4):582–599, 2023.

[69] José Bacelar Almeida, Maria João Frade, Jorge Sousa Pinto, and Simão Melo de Sousa. An overview of formal methods tools and techniques. *Rigorous Software Development: An Introduction to Program Verification*, pages 15–44, 2011.

[70] Robert M Hierons, Kirill Bogdanov, Jonathan P Bowen, Rance Cleaveland, John Derrick, Jeremy Dick, Marian Gheorghe, Mark Harman, Kalpesh Kapoor, Paul Krause, et al. Using formal specifications to support testing. *ACM Computing Surveys (CSUR)*, 41(2):1–76, 2009.

[71] M-C Gaudel. Formal specification techniques. In *Proceedings of 16th International Conference on Software Engineering*, pages 223–227. IEEE, 1994.

[72] Jeannette M Wing. A specifier's introduction to formal methods. *Computer*, 23(9):8–22, 1990.

[73] Robert W Floyd. Assigning meanings to programs. In *Program Verification: Fundamental Issues in Computer Science*, pages 65–81. Springer, 1993.

[74] VS Alagar, K Periyasamy, VS Alagar, and K Periyasamy. Vienna development method. *Specification of Software Systems*, pages 405–459, 2011.

[75] Kristin Y Rozier. Linear temporal logic symbolic model checking. *Computer Science Review*, 5(2):163–203, 2011.

[76] Hillel Kugler, David Harel, Amir Pnueli, Yuan Lu, and Yves Bontemps. Temporal logic for scenario-based specifications. In *Tools and Algorithms for the Construction and Analysis of Systems: 11th International Conference, TACAS 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005. Proceedings 11*, pages 445–460. Springer, 2005.

[77] Peter Bulychev, Alexandre David, Kim Gulstrand Larsen, Marius Mikučionis, Danny Bøgsted Poulsen, Axel Legay, and Zheng Wang. Uppaal-smc: Statistical model checking for priced timed automata. *arXiv preprint arXiv:1207.1272*, 2012.

[78] David Parker. Verification of probabilistic real-time systems. *Proc. 2013 Real-time Systems Summer School (ETR'13)*, 2013.

[79] Stephen A Cook. The complexity of theorem-proving procedures. In *Logic, automata, and computational complexity: The works of Stephen A. Cook*, pages 143–152. 2023.

[80] Yves Bertot and Pierre Castéran. *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*. Springer Science & Business Media, 2013.

[81] Lawrence C Paulson. *Isabelle: A generic theorem prover*. Springer, 1994.

[82] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[83] Edmund M Clarke, William Klieber, Miloš Nováček, and Paolo Zuliani. Model checking and the state explosion problem. In *LASER Summer School on Software Engineering*, pages 1–30. Springer, 2011.

[84] Junaid Qadir and Osman Hasan. Applying formal methods to networking: theory, techniques, and applications. *IEEE Communications Surveys & Tutorials*, 17(1):256–291, 2014.

[85] Jerry R Burch, Edmund M Clarke, Kenneth L McMillan, David L Dill, and Lain-Jinn Hwang. Symbolic model checking: 1020 states and beyond. *Information and computation*, 98(2):142–170, 1992.

[86] Armin Biere, Alessandro Cimatti, Edmund Clarke, and Yunshan Zhu. Symbolic model checking without bdds. In *Tools and Algorithms for the Construction and Analysis of Systems: 5th International Conference, TACAS'99 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS'99 Amsterdam, The Netherlands, March 22–28, 1999 Proceedings 5*, pages 193–207. Springer, 1999.

[87] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism: probabilistic model checking for performance and reliability analysis. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):40–45, 2009.

[88] Koushik Sen, Mahesh Viswanathan, and Gul Agha. Statistical model checking of black-box probabilistic systems. In Rajeev Alur and Doron Peled, editors, *CAV*, pages 202–215, 2004.

[89] Alexandre David, Kim G Larsen, Axel Legay, Marius Mikučionis, and Zheng Wang. Time for statistical model checking of real-time systems. In *Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings 23*, pages 349–355. Springer, 2011.

[90] Alexandre David, Kim G Larsen, Axel Legay, Marius Mikučionis, Danny Bøgsted Poulsen, Jonas Van Vliet, and Zheng Wang. Statistical model checking for networks of priced timed automata. In *Formal Modeling and Analysis of Timed Systems: 9th International Conference, FORMATS 2011, Aalborg, Denmark, September 21-23, 2011. Proceedings 9*, pages 80–96. Springer, 2011.

[91] Paolo Ballarini, Hilal Djafri, Marie Duflot, Serge Haddad, and Nihal Pekergin. Cosmos: a statistical model checker for the hybrid automata stochastic logic. In *2011 Eighth International Conference on Quantitative Evaluation of SysTems*, pages 143–144. IEEE, 2011.

[92] Musab AlTurki and José Meseguer. Pvesta: A parallel statistical model checking and quantitative analysis tool. In *International Conference on Algebra and Coalgebra in Computer Science*, pages 386–392. Springer, 2011.

[93] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking in dense real-time. *Information and computation*, 104(1):2–34, 1993.

[94] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal aspects of computing*, 6:512–535, 1994.

[95] Majda Moussa. *Vérification et configuration automatiques de pare-feux par Model Checking et synthèse de contrôleur*. PhD thesis, École Polytechnique de Montréal, 2014.

[96] Rajeev Alur and David Dill. Automata for modeling real-time systems. In *Automata, Languages and Programming: 17th International Colloquium Warwick University, England, July 16–20, 1990 Proceedings 17*, pages 322–335. Springer, 1990.

[97] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[98] Marta Kwiatkowska, Gethin Norman, and Jeremy Sproston. *Symbolic Computation of Maximal Probabilisti Reachability*, pages 169–183. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[99] James Lyle Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

[100] M. Kamath and N. Viswanadham. Applications of Petri net based models in the modelling and analysis of flexible manufacturing systems. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 312–317, April 1986.

[101] Claude Girault and Rudiger Valk. *Petri Nets for System Engineering: A Guide to Modeling, Verification, and Applications*. Springer-Verlag, Berlin, Heidelberg, 2001.

[102] Irina A Lomazova. Nested petri nets—a formalism for specification and verification of multi-agent distributed systems. *Fundamenta informaticae*, 43(1-4):195–214, 2000.

[103] Jiafeng Zhang, Mohamed Khalgui, Zhiwu Li, Olfa Mosbahi, and Abdulrahman M Al-Ahmari. R-tnces: A novel formalism for reconfigurable discrete event control systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(4):757–772, 2013.

[104] H-M Hanisch, J Thieme, Arndt Luder, and O Wienhold. Modeling of plc behavior by means of timed net condition/event systems. In *1997 IEEE 6th International Conference on Emerging Technologies and Factory Automation Proceedings, EFTA'97*, pages 391–396. IEEE, 1997.

[105] Rüdiger Valk. Self-modifying nets, a natural extension of Petri nets. In *International Colloquium on Automata, Languages, and Programming*, pages 464–476. Springer, 1978.

[106] Rüdiger Valk. On the computational power of extended Petri nets. In *International Symposium on Mathematical Foundations of Computer Science*, pages 526–535. Springer, 1978.

[107] Rüdiger Valk. Petri nets as token objects. In *International Conference on Application and Theory of Petri Nets*, pages 1–24. Springer, 1998.

[108] Rüdiger Valk. Concurrency in communicating object Petri nets. In *Concurrent Object-Oriented Programming and Petri Nets*, pages 164–195. Springer, 2001.

[109] Rüdiger Valk. Object Petri nets. In *Lectures on Concurrency and Petri Nets*, pages 819–848. Springer, 2004.

[110] Hartmut Ehrig and Julia Padberg. *Graph Grammars and Petri Net Transformations*, pages 496–536. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[111] Marisa Llorens and Javier Oliver. Introducing structural dynamic changes in Petri nets: Marked-controlled reconfigurable nets. In *Proc. of Second International Conference on Automated Technology for Verification and Analysis*, pages 310–323. Springer, 2004.

[112] Marisa Llorens and Javier Oliver. Structural and dynamic changes in concurrent systems: Reconfigurable Petri nets. *IEEE Transactions on Computers*, 53(9):1147–1158, 2004.

[113] Jun Li, Xianzhong Dai, and Zhengda Meng. Improved net rewriting systems-based rapid reconfiguration of Petri net logic controllers. In *31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005.* IEEE, 2005.

[114] Jun Li, Xianzhong Dai, and Zhengda Meng. Improved net rewriting system-based approach to model reconfiguration of reconfigurable manufacturing systems. *The International Journal of Advanced Manufacturing Technology*, 37(11-12):1168–1189, 2008.

[115] Jun Li, Xianzhong Dai, Zhengda Meng, and Libo Xu. Improved net rewriting system-extended Petri net supporting dynamic changes. *Journal of Circuits, Systems, and Computers*, 17(06):1027–1052, 2008.

[116] J. Li, X. Dai, and Z. Meng. Automatic reconfiguration of Petri net controllers for reconfigurable manufacturing systems with an improved net rewriting system-based approach. *IEEE Transactions on Automation Science and Engineering*, 6(1):156–167, 2009.

[117] Jun Li, Xianzhong Dai, Zhengda Meng, Jianping Dou, and Xianping Guan. Rapid design and reconfiguration of Petri net models for reconfigurable manufacturing cells with improved net rewriting systems and activity diagrams. *Computers & Industrial Engineering*, 57(4):1431–1451, 2009.

[118] Eric Badouel, Marisa Llorens, and Javier Oliver. Modeling concurrent systems: Reconfigurable nets. In *Proc. PDPTA*, pages 1568–1574, 2003.

[119] Zbigniew Michalewicz and Marc Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary computation*, 4(1):1–32, 1996.

[120] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia tools and applications*, 80:8091–8126, 2021.

[121] Gan Kim Soon, Tan Tse Guan, Chin Kim On, Rayner Alfred, and Patricia Anthony. A comparison on the performance of crossover techniques in video game. In *2013 IEEE international conference on control system, computing and engineering*, pages 493–498. IEEE, 2013.

[122] David E Goldberg and Robert Lingle. Alleles, loci, and the traveling salesman problem. In *Proceedings of the first international conference on genetic algorithms and their applications*, pages 154–159. Psychology Press, 2014.

[123] Shubhra Sankar Ray, Sanghamitra Bandyopadhyay, and Sankar K Pal. New operators of genetic algorithms for traveling salesman problem. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, pages 497–500. IEEE, 2004.

[124] Kazi Shah Nawaz Ripon, Nazmul H Siddique, and Jim Torresen. Improved precedence preservation crossover for multi-objective job shop scheduling problem. *Evolving Systems*, 2:119–129, 2011.

[125] Forbes J Burkowski. Shuffle crossover and mutual information. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1574–1580. IEEE, 1999.

[126] Siew Mooi Lim, Abu Bakar Md Sultan, Md Nasir Sulaiman, Aida Mustapha, and Kuan Yew Leong. Crossover and mutation operators of genetic algorithms. *International journal of machine learning and computing*, 7(1):9–12, 2017.

[127] Devasenathipathi N Mudaliar and Nilesh K Modi. Unraveling travelling salesman problem by genetic algorithm using m-crossover operator. In *2013 International Conference on Signal Processing, Image Processing & Pattern Recognition*, pages 127–130. IEEE, 2013.

[128] Khalid Jebari, Mohammed Madiafi, et al. Selection methods for genetic algorithms. *International Journal of Emerging Sciences*, 3(4):333–344, 2013.

[129] Carlos M Fonseca and Peter J Fleming. Multiobjective genetic algorithms. In *IEE colloquium on genetic algorithms for control systems engineering*, pages 6–1. Iet, 1993.

[130] MC Bhuvaneswari and G Subashini. Introduction to multi-objective evolutionary algorithms. *Application of Evolutionary Algorithms for Multi-objective Optimization in VLSI and Embedded Systems*, pages 1–20, 2015.

[131] Kalyanmoy Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary computation*, 7(3):205–230, 1999.

[132] Abdullah Konak, David W Coit, and Alice E Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability engineering & system safety*, 91(9):992–1007, 2006.

[133] J David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the first international conference on genetic algorithms and their applications*, pages 93–100. Psychology Press, 2014.

[134] Jeffrey Horn, Nicholas Nafpliotis, and David E Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*, pages 82–87. Ieee, 1994.

[135] Nidamarthi Srinivas and Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

[136] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr 2002.

[137] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[138] Joshua D Knowles and David W Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary computation*, 8(2):149–172, 2000.

[139] Tadahiko Murata, Hisao Ishibuchi, et al. Moga: multi-objective genetic algorithms. In *IEEE international conference on evolutionary computation*, volume 1, pages 289–294. IEEE Piscataway, 1995.

[140] Haiming Lu and Gary G Yen. Rank-density-based multiobjective genetic algorithm and benchmark test function study. *IEEE transactions on evolutionary computation*, 7(4):325–343, 2003.

[141] Manel Houimli, Laid Kahloul, and Siham Benaoune. Performance analysis of internet of things application layer protocol. In *Lecture Notes in Real-Time Intelligent Systems*, pages 225–234. Springer, 2019.

[142] Manel Houimli and Laid Kahloul. Formal verification of collision free mobility adaptive protocol for wireless sensor networks. 2017.

[143] S. A. Ajith Kumar and Kent I. F. Simonsen. Towards a model-based development approach for wireless sensor-actuator network protocols. In *Proceedings of the 4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*, CyPhy '14, pages 35–39, New York, NY, USA, 2014. ACM.

[144] Y. Hammal, J. Ben-Othman, L. Mokdad, and A. Abdelli. Formal modeling and verification of an enhanced variant of the ieee 802.11 csma/ca protocol. *Journal of Communications and Networks*, 16(4):385–396, Aug 2014.

[145] Zohra Hmidi, Laïd Kahloul, Benharzallah Saber, and Cherifa Othmane. Statistical model checking of CSMA/CA in wsns. In *Proceedings of the 10th Workshop on Verification and Evaluation of Computer and Communication System, VECoS 2016, Tunis, Tunisia, October 6-7, 2016.*, pages 27–42, 2016.

[146] Zhi Chen, Ya Peng, and Wenjing Yue. Modeling and analyzing csma/ca protocol for energy-harvesting wireless sensor networks. *International Journal of Distributed Sensor Networks*, 11(9):257157, 2015.

[147] Z. Li, Y. Liu, M. Li, J. Wang, and Z. Cao. Exploiting ubiquitous data collection for mobile users in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 24(2):312–326, Feb 2013.

[148] C. Tunca, S. Isik, M. Y. Donmez, and C. Ersoy. Ring routing: An energy-efficient routing protocol for wireless sensor networks with a mobile sink. *IEEE Transactions on Mobile Computing*, 14(9):1947–1960, Sept 2015.

[149] J. He, P. Cheng, J. Chen, L. Shi, and R. Lu. Time synchronization for random mobile sensor networks. *IEEE Transactions on Vehicular Technology*, 63(8):3935–3946, Oct 2014.

[150] M. J. Nene, R. S. Deodhar, and L. M. Patnaik. Algorithm for autonomous reorganization of mobile wireless camera sensor networks to improve coverage. *IEEE Sensors Journal*, 15(8):4428–4441, Aug 2015.

[151] Sheeraz A Alvi, Ghalib A Shah, and Waqar Mahmood. Energy efficient green routing protocol for internet of multimedia things. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on*, pages 1–6. IEEE, 2015.

[152] Adarsh Kumar, Krishna Gopal, and Alok Aggarwal. Simulation and analysis of authentication protocols for mobile internet of things (miot). In *Parallel, Distributed and Grid Computing (PDGC), 2014 International Conference on*, pages 423–428. IEEE, 2014.

[153] Antonio De Rubertis, Luca Mainetti, Vincenzo Mighali, Luigi Patrono, Ilaria Sergi, Maria Laura Stefanizzi, and Stefano Pascali. Performance evaluation of end-to-end security protocols in an internet of things. In *Software, Telecommunications and Computer Networks (SoftCOM), 2013 21st International Conference on*, pages 1–6. IEEE, 2013.

[154] Quan Le, Thu Ngo-Quynh, and Thomaz Magedanz. Rpl-based multipath routing protocols for internet of things on wireless sensor networks. In *Advanced Technologies for Communications (ATC), 2014 International Conference on*, pages 424–429. IEEE, 2014.

[155] Zoran B Babovic, Jelica Protic, and Veljko Milutinovic. Web performance evaluation for internet of things applications. *IEEE Access*, 4:6974–6992, 2016.

[156] Matteo Collina, Marco Bartolucci, Alessandro Vanelli-Coralli, and Giovanni Emanuale Corazza. Internet of things application layer protocol analysis over error and delay prone links. In *Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC), 2014 7th*, pages 398–404. IEEE, 2014.

[157] Leila Abidi, Christophe Cérin, and Sami Evangelista. A petri-net model for the publish-subscribe paradigm and its application for the verification of the bonjourgrid middleware. In *Services Computing (SCC), 2011 IEEE International Conference on*, pages 496–503. IEEE, 2011.

[158] K. Saghar, W. Henderson, D. Kendall, and A. Bouridane. Formal modelling of a robust wireless sensor network routing protocol. In *2010 NASA/ESA Conference on Adaptive Hardware and Systems*, pages 281–288, June 2010.

[159] Maissa Elleuch, Osman Hasan, Sofiène Tahar, and Mohamed Abid. Formal probabilistic analysis of a wireless sensor network for forest fire detection. In Adel Bouhoula, Tetsuo Ida, and Fairouz Kamareddine, editors, *SCSS*, volume 122 of *EPTCS*, pages 1–9, 2012.

[160] Juan Antonio Cordero. A probabilistic study of the delay caused by jittering in wireless flooding. *Wireless Personal Communications*, 73(3):415–439, Dec 2013.

[161] Tien-Shin Ho and Kwang-Cheng Chen. Performance analysis of ieee 802.11 csma/ca medium access control protocol. In *Personal, Indoor and Mobile Radio Communications, 1996. PIMRC'96., Seventh IEEE International Symposium on*, volume 2, pages 407–411 vol.2, Oct 1996.

[162] Hongqiang Zhai, Younggoo Kwon, and Yuguang Fang. Performance analysis of ieee 802.11 mac protocols in wireless lans. *Wireless Communications and Mobile Computing*, 4(8):917–931, 2004.

[163] Fengling Zhang, Lei Bu, Linzhang Wang, Jianhua Zhao, Xin Chen, Tian Zhang, and Xuandong Li. Modeling and evaluation of wireless sensor network protocols by stochastic timed automata. *Electronic Notes in Theoretical Computer Science*, 296:261 – 277, 2013. Proceedings the Sixth International Workshop on the Practical Application of Stochastic Modelling (PASM) and the Eleventh International Workshop on Parallel and Distributed Methods in Verification (PDMC).

[164] Paolo Ballarini and Alice Miller. Model checking medium access control for sensor networks. In *Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods*, (ISoLA'06), pages 255–262, 2006.

[165] Kurt Jensen, Lars Michael Kristensen, and Lisa Wells. Coloured petri nets and cpn tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, 9(3):213–254, Jun 2007.

[166] Marta Kwiatkowska, Gethin Norman, and Jeremy Sproston. *Probabilistic Model Checking of the IEEE 802.11 Wireless Local Area Network Protocol*, pages 169–187. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[167] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, pages 138–149, New York, NY, USA, 2003. ACM.

[168] Cinzia Bernardeschi, Paolo Masci, and Holger Pfeifer. *Analysis of Wireless Sensor Network Protocols in Dynamic Scenarios*, pages 105–119. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[169] C. Zhang and M. Zhou. A stochastic petri net-approach to modeling and analysis of ad hoc network. In *International Conference on Information Technology: Research and Education, 2003. Proceedings. ITRE2003.*, pages 152–156, Aug 2003.

[170] Stephen Kent and Karen Seo. Security architecture for the internet protocol. Technical report, 2005.

[171] Eric Rescorla and Nagendra Modadugu. Datagram transport layer security version 1.2. Technical report, 2012.

[172] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.

[173] Jorge Granjal, Edmundo Monteiro, and Jorge Sá Silva. Security for the internet of things: a survey of existing protocols and open research issues. *IEEE Communications Surveys & Tutorials*, 17(3):1294–1312, 2015.

[174] Roberto Baldoni, Mariangela Contenti, Sara Tucci Piergiovanni, and Antonino Virgillito. Modeling publish/subscribe communication systems: towards a formal approach. In *Object-Oriented Real-Time Dependable Systems, 2003.(WORDS 2003). Proceedings of the Eighth International Workshop on*, pages 304–311. IEEE, 2003.

[175] Heithem Abbes, Christophe Cérin, and Mohamed Jemni. Bonjourgrid as a decentralised job scheduler. In *2008 IEEE Asia-Pacific Services Computing Conference*, pages 89–94. IEEE, 2008.

[176] Ahmad Naseem Alvi, Syed Saud Naqvi, Safdar Hussain Bouk, Nadeem Javaid, Umar Qasim, and Zahoor Ali Khan. Evaluation of slotted csma/ca of ieee 802.15. 4. In *2012 seventh international conference on broadband, wireless computing, communication and applications*, pages 391–396. IEEE, 2012.

[177] What is uml? `http://www.uml.org/`. Accessed: 2017-06-29.

[178] Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikuăionis, and Danny BOgsted Poulsen. Uppaal smc tutorial. *Int. J. Softw. Tools Technol. Transf.*, 17(4):397–415, August 2015.

[179] Manfredi Bruccoleri, Zbigniew J Pasek, and Yoram Koren. Operation management in reconfigurable manufacturing systems: Reconfiguration for error handling. *International Journal of Production Economics*, 100(1):87–100, 2006.

[180] M Maniraj, V Pakkirisamy, and R Jeyapaul. An ant colony optimization–based approach for a single-product flow-line reconfigurable manufacturing systems. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 231(7):1229–1236, 2017.

[181] Xuemei Liu, Jiawei Chen, and Aiping Li. Optimisation of line configuration and balancing for reconfigurable transfer lines considering demand uncertainty. *International Journal of Production Research*, pages 1–23, 2019.

[182] Rajesh Pansare, Gunjan Yadav, and Madhukar R Nagare. Development of a structural framework to improve reconfigurable manufacturing system adoption in the manufacturing industry. *International Journal of Computer Integrated Manufacturing*, 36(3):349–380, 2023.

[183] Alessia Napoleone, Ann-Louise Andersen, Thomas Ditlev Brunoe, and Kjeld Nielsen. Towards human-centric reconfigurable manufacturing systems: Literature review of reconfigurability enablers for reduced reconfiguration effort and classification frameworks. *Journal of Manufacturing Systems*, 67:23–34, 2023.

[184] Jelena Milisavljevic-Syed, Jiahong Li, and Hanbing Xia. Realisation of responsive and sustainable reconfigurable manufacturing systems. *International Journal of Production Research*, 62(8):2725–2746, 2024.

[185] Rebeca Arista, Fernando Mas, Domingo Morales-Palma, and Carpoforo Vallellano. An ontology-based engineering methodology applied to aerospace reconfigurable manufacturing systems design. *International Journal of Production Research*, 62(6):2286–2304, 2024.

[186] Rasmus Andersen, Alessia Napoleone, Ann-Louise Andersen, Thomas Ditlev Brunoe, and Kjeld Nielsen. A systematic methodology for changeable and reconfigurable manufacturing systems development. *Journal of Manufacturing Systems*, 74:449–462, 2024.

[187] Bohan Leng, Shuo Gao, Tangbin Xia, Ershun Pan, Joachim Seidelmann, Hao Wang, and Lifeng Xi. Digital twin monitoring and simulation integrated platform for reconfigurable manufacturing systems. *Advanced Engineering Informatics*, 58:102141, 2023.

[188] Iwona Grobelna, Remigiusz Wiśniewski, Michał Grobelny, and Monika Wiśniewska. Design and verification of real-life processes with application of petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(11):2856–2869, 2017.

[189] Jun Li, Xianghu Meng, MengChu Zhou, and Xianzhong Dai. A two-stage approach to path planning and collision avoidance of multibridge machining systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(7):1039–1049, 2017.

[190] Dimitri Lefebvre, Sara Rachidi, Edouard Leclercq, and Yoann Pigné. Diagnosis of structural and temporal faults for k-bounded non-markovian stochastic petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[191] Jianhong Ye, MengChu Zhou, Zhiwu Li, and Abdulrahman Al-Ahmari. Structural decomposition and decentralized control of petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(8):1360–1369, 2018.

[192] Jun Li, Xiaolong Yu, and MengChu Zhou. Analysis of unbounded petri net with lean reachability trees. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[193] Tadao Murata. Petri nets: Properties, analysis and applications. *Proc. IEEE*, 77(4):541–580, 1989.

[194] Mu Der Jeng. Petri nets for modeling automated manufacturing systems with error recovery. *IEEE Transactions on Robotics and Automation*, 13(5):752–760, 1997.

[195] Xiuli Meng. Modeling of reconfigurable manufacturing systems based on colored timed object-oriented petri nets. *Journal of Manufacturing Systems*, 29(2):81–90, 2010.

[196] NaiQi Wu and MengChu Zhou. Intelligent token petri nets for modelling and control of reconfigurable automated manufacturing systems with dynamical changes. *Transactions of the Institute of Measurement and Control*, 33(1):9–29, 2011.

[197] Lianfeng Zhang and Brian Rodrigues. Modelling reconfigurable manufacturing systems with coloured timed petri nets. *International journal of production research*, 47(16):4569–4591, 2009.

[198] Enrico Biermann and Tony Modica. Independence analysis of firing and rule-based net transformations in reconfigurable object nets. *Electronic Communications of the EASST*, 10, 2008.

[199] Ulrike Prange, Hartmut Ehrig, Kathrin Hoffmann, and Julia Padberg. *Transformations in Reconfigurable Place/Transition Systems*, pages 96–113. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[200] Eric Badouel and Javier Oliver. *Reconfigurable nets, a class of high level Petri nets supporting dynamic changes within workflow systems*. PhD thesis, INRIA, 1998.

[201] Jun Li, Xianzhong Dai, Zhengda Meng, Jianping Dou, and Xianping Guan. Rapid design and reconfiguration of petri net models for reconfigurable manufacturing cells with improved net rewriting systems and activity diagrams. *Computers & Industrial Engineering*, 57(4):1431–1451, 2009.

[202] Huixia Liu, Keyi Xing, MengChu Zhou, Libin Han, and Feng Wang. Transition cover-based design of petri net controllers for automated manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(2):196–208, 2014.

[203] Jiafeng Zhang, Mohamed Khalgui, Zhiwu Li, Georg Frey, Olfa Mosbahi, and Hela Ben Salah. Reconfigurable coordination of distributed discrete event control systems. *IEEE Trans. Contr. Sys. Techn.*, 23(1):323–330, 2015.

[204] Bo Huang, Hang Zhu, Gongxuan Zhang, and Xianling Lu. On further reduction of constraints in “nonpure petri net supervisors for optimal deadlock control of flexible manufacturing systems”. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(3):542–543, 2015.

[205] Bo Huang, MengChu Zhou, GongXuan Zhang, Ahmed Chiheb Ammari, Ahmed Alabdulwahab, and Ayman G Fayoumi. Lexicographic multiobjective integer programming for optimal and structurally minimal petri net supervisors of automated manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(11):1459–1470, 2015.

[206] JianChao Luo, KeYi Xing, MengChu Zhou, XiaoLing Li, and XinNian Wang. Deadlock-free scheduling of automated manufacturing systems using petri nets and hybrid heuristic search. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(3):530–541, 2015.

[207] Huixia Liu, Keyi Xing, Weimin Wu, MengChu Zhou, and Hailin Zou. Deadlock prevention for flexible manufacturing systems via controllable siphon basis of petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(3):519–529, 2015.

[208] Yanxiang Feng, Keyi Xing, Zhenxin Gao, and Yunchao Wu. Transition cover-based robust petri net controllers for automated manufacturing systems with a type of unreliable resources. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2016.

[209] Olatunde T Baruwa, Miquel Angel Piera, and Antoni Guasch. Deadlock-free scheduling method for flexible manufacturing systems based on timed colored petri nets and anytime heuristic search. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(5):831–846, 2015.

[210] QingHua Zhu, MengChu Zhou, Yan Qiao, and NaiQi Wu. Petri net modeling and scheduling of a close-down process for time-constrained single-arm cluster tools. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2016.

[211] Laid Kahloul, Samir Bourekkache, and Karim Djouani. Designing reconfigurable manufacturing systems using reconfigurable object petri nets. *International Journal of Computer Integrated Manufacturing*, 29(8):889–906, 2016.

[212] O. Khlifi, O. Mosbahi, M. Khalgui, G. Frey, and Z. Li. Modeling, simulation and verification of probabilistic reconfigurable discrete-event systems under energy and memory constraints. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, Jul 2018.

[213] S. Guellouz, A. Benzina, M. Khalgui, G. Frey, Z. Li, and V. Vyatkin. Designing efficient reconfigurable control systems using iec61499 and symbolic model checking. *IEEE Transactions on Automation Science and Engineering*, pages 1–15, 2018.

[214] Samir Tigane, Laid Kahloul, Nadia Hamani, Mohamed Khalgui, and Masood Ashraf Ali. On quantitative properties preservation in reconfigurable generalized stochastic petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2022.

[215] Samir Tigane, Fayçal Guerrouf, and Laid Kahloul. A gspn-based formalism under infinite-server semantics for reconfigurable wanets. *Computing*, pages 1–29, 2023.

[216] N Ismail, F Musharavati, ASM Hamouda, and AR Ramli. Manufacturing process planning optimisation in reconfigurable multiple parts flow lines. *Journal of achievements in materials and manufacturing engineering*, 31(2):671–677, 2008.

[217] Abderrahmane Bensmaine, Mohammed Dahane, and Lyes Benyoucef. A non-dominated sorting genetic algorithm based approach for optimal machines selection in reconfigurable manufacturing environment. *Computers & Industrial Engineering*, 66(3):519–524, 2013.

[218] Anuch Chaube, Lyès Benyoucef, and Manoj Kumar Tiwari. An adapted nsga-2 algorithm based dynamic process plan generation for a reconfigurable manufacturing system. *Journal of Intelligent Manufacturing*, 23(4):1141–1155, 2012.

[219] Wencai Wang and Yoram Koren. Scalability planning for reconfigurable manufacturing systems. *Journal of Manufacturing Systems*, 31(2):83–91, 2012.

[220] Abderrahmane Bensmaine, Mohamed Dahane, and Lyes Benyoucef. A new heuristic for integrated process planning and scheduling in reconfigurable manufacturing systems. *International Journal of Production Research*, 52(12):3583–3594, 2014.

[221] Xie Xiaowen, Zheng Beirong, and Xue Wei. Configuration optimization method of reconfigurable manufacturing systems. *Research Journal of Applied Sciences, Engineering and Technology*, 6(8):389–1393, 2013.

[222] Jianping Dou, Jun Li, and Chun Su. Bi-objective optimization of integrating configuration generation and scheduling for reconfigurable flow lines using nsga-ii. *The International Journal of Advanced Manufacturing Technology*, 86(5-8):1945–1962, 2016.

[223] Yousra Hafidi, Laïd Kahloul, Mohamed Khalgui, Zhiwu Li, Khalid Alnowibet, and Ting Qu. On methodology for the verification of reconfigurable timed net condition/event systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, (99):1–15, 2018.

[224] Wafa Lakhdhar, Rania Mzid, Mohamed Khalgui, Zhiwu Li, Georg Frey, and Abdulrahman Al-Ahmari. Multiobjective optimization approach for a portable development of reconfigurable real-time systems: From specification to implementation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[225] Dashuang Li, Chaoyong Zhang, Guangdong Tian, Xinyu Shao, and Zhiwu Li. Multiobjective program and hybrid imperialist competitive algorithm for the mixed-model two-sided assembly lines subject to multiple constraints. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(1):119–129, 2018.

[226] Sihan Huang, Jiaxin Tan, Yuqian Lu, Shokraneh K Moghaddam, Guoxin Wang, and Yan Yan. A multi-objective joint optimisation method for simultaneous part family formation and configuration design in delayed reconfigurable manufacturing system (d-rms). *International Journal of Production Research*, pages 1–18, 2023.

[227] Wei Niu, Jun-qing Li, Hui Jin, Rui Qi, and Hong-yan Sang. Bi-objective optimization using an improved nsga-ii for energy-efficient scheduling of a distributed assembly blocking flowshop. *Engineering Optimization*, 55(5):719–740, 2023.

[228] Mingkun Yang, Yuhang Zhang, Chao Ai, Guishan Yan, and Wenguang Jiang. Multi-objective optimisation of k-shape notch multi-way spool valve using cfd analysis, discharge area parameter model, and nsga-ii algorithm. *Engineering Applications of Computational Fluid Mechanics*, 17(1):2242721, 2023.

[229] Mariagrazia Dotoli, Maria Pia Fanti, Alessandro Giua, and Carla Seatzu. First-order hybrid petri nets. an application to distributed manufacturing systems. *Nonlinear Analysis: Hybrid Systems*, 2(2):408–430, 2008.

[230] Fabio Balduzzi, Alessandro Giua, and Carla Seatzu. Modelling and simulation of manufacturing systems with first-order hybrid petri nets. *International Journal of Production Research*, 39(2):255–282, 2001.

[231] Graziana Cavone, Mariagrazia Dotoli, and Carla Seatzu. Management of intermodal freight terminals by first-order hybrid petri nets. *IEEE Robotics and Automation Letters*, 1(1):2–9, 2015.

[232] G Cavone, M Dotoli, N Epicoco, M Franceschelli, and C Seatzu. Hybrid petri nets to re-design low-automated production processes: the case study of a sardinian bakery. *IFAC-PapersOnLine*, 51(7):265–270, 2018.

[233] Armin Zimmermann, Diego Rodriguez, and M Silva. A two phase optimization method for petri net models of manufacturing systems. *Journal of Intelligent Manufacturing*, 12(5-6):409–420, 2001.

[234] Gonzalo Mejía and Carlos Montoya. Scheduling manufacturing systems with blocking: a petri net approach. *International Journal of Production Research*, 47(22):6261–6277, 2009.

[235] Gašper Mušič. Generation of feasible petri net based scheduling problem solutions. *IFAC-PapersOnLine*, 48(1):856 – 861, 2015. 8th Vienna International Conferenceon Mathematical Modelling.

[236] Jeongsun Ahn and Hyun-Jung Kim. A branch and bound algorithm for scheduling of flexible manufacturing systems. *IEEE Transactions on Automation Science and Engineering*, 2023.

[237] Abdulmajeed Dabwan, Husam Kaid, Abdulrahman Al-Ahmari, Khaled N Alqahtani, and Wadea Ameen. An internet-of-things-based dynamic scheduling optimization method for unreliable flexible manufacturing systems under complex operational conditions. *Machines*, 12(3):192, 2024.

[238] Fabio Balduzzi, Alessandro Giua, and Giuseppe Menga. First-order hybrid petri nets: a model for optimization and control. *IEEE transactions on robotics and automation*, 16(4):382–399, 2000.

[239] Hung-We Wen, Li-Chen Fu, and Shih-Shinh Huang. Modeling, scheduling, and prediction in wafer fabrication systems using queueing petri net and genetic algorithm. In *Proc. Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 3559–3564. IEEE, 2001.

[240] Kazuhiro Saitou, Samir Malpathak, and Helge Qvam. Robust design of flexible manufacturing systems using, colored petri net and genetic algorithm. *Journal of intelligent manufacturing*, 13(5):339–351, 2002.

[241] Napalkova Liana, Merkuryeva Galina, and Piera Miquel Angel. Development of genetic algorithm for solving scheduling tasks in fms with coloured petri nets. In *Proc. of the International Mediterranean Modeling Multiconference*, 2006.

[242] C-F Chien and C-H Chen. Using genetic algorithms (ga) and a coloured timed petri net (ctpn) for modelling the optimization-based schedule generator of a generic production scheduling system. *International Journal of Production Research*, 45(8):1763–1789, 2007.

[243] Gonzalo Mejía, Carlos Montoya, Julian Cardona, and Ana Lucía Castro. Petri nets and genetic algorithms for complex manufacturing systems scheduling. *International Journal of Production Research*, 50(3):791–803, 2012.

[244] Anita Gudelj, Danko Kezić, and Stjepan Vidačić. Planning and optimization of agv jobs by petri net and genetic algorithm. *Journal of Information and Organizational Sciences*, 36(2):99–122, 2012.

[245] Albert WL Yao and YM Pan. A petri nets and genetic algorithm based optimal scheduling for job shop manufacturing systems. In *Proc. System Science and Engineering (ICSSE), 2013 International Conference on*, pages 99–104. IEEE, 2013.

[246] Marco S Nobile, Daniela Besozzi, Paolo Cazzaniga, and Giancarlo Mauri. The foundation of evolutionary petri nets. In *Proc. BioPPN at Petri Nets*, pages 60–74, 2013.

[247] Juan Pablo Caballero-Villalobos, Gonzalo Enrique Mejía-Delgadillo, and Rafael Guillermo García-Cáceres. Scheduling of complex manufacturing systems with petri nets and genetic algorithms: a case on plastic injection moulds. *The International Journal of Advanced Manufacturing Technology*, 69(9-12):2773–2786, 2013.

[248] D Sreeramulu, Y Sagar, P Suman, and A Satish Kumar. Integration of process planning and scheduling of a manufacturing systems using petri nets and genetic algorithm. *Indian Journal of Science and Technology*, 9(41), 2016.

[249] Li Tingpeng, Wang Nantian, Li Yue, and Qian Yanling. An improved optimization algorithm for timed petri net based on genetic algorithm. In *Proc. 3rd International Conference on Materials Engineering, Manufacturing Technology and Control (ICMEMTC 2016)*, pages 956–966. Atlantis Press, 2016.

[250] Yain-Whar Si, Veng-Ian Chan, Marlon Dumas, and Defu Zhang. A petri nets based generic genetic algorithm framework for resource optimization in business processes. *Simulation Modelling Practice and Theory*, 86:72–101, 2018.

[251] Chen Chen, Chan Gu, and Hesuan Hu. Optimal supervisor simplification in ams based on petri nets and genetic algorithm. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 1757–1764. IEEE, 2021.

[252] Gašper Mušič. Pn-ga based optimization of flexible job shop schedules. *IFAC-PapersOnLine*, 55(20):517–522, 2022.

[253] Gaetano Volpe, Agostino Marcello Mangini, and Maria Pia Fanti. Job shop sequencing in manufacturing plants by timed coloured petri nets and particle swarm optimization. *IFAC-PapersOnLine*, 55(28):350–355, 2022.

[254] Xingkai WANG, Weimin WU, Zichao XING, Tingqi ZHANG, and Haoyi NIU. A look-ahead agv scheduling algorithm with processing sequence conflict-free for a no-buffer assembly line. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, 17(5):JAMDSM0063–JAMDSM0063, 2023.

[255] Aiping Li and Nan Xie. A robust scheduling for reconfigurable manufacturing system using petri nets and genetic algorithm. In *Proc. Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, volume 2, pages 7302–7306. IEEE, 2006.

[256] Fu-Shiung Hsieh. Design of scalable agent-based reconfigurable manufacturing systems with petri nets. *International Journal of Computer Integrated Manufacturing*, 31(8):748–759, 2018.

[257] Fu-Shiung Hsieh. A dynamic context-aware workflow management scheme for cyber-physical systems based on multi-agent system architecture. *Applied Sciences*, 11(5):2030, 2021.

[258] Manel Houimli, Laid Kahloul, and Mohamed Khalgui. On formal modeling, analysis and optimization of reconfigurable manufacturing systems. *International Journal of Computer Integrated Manufacturing*, pages 1–31, 2024.

[259] Enrico Biermann, Claudia Ermel, Frank Hermann, and Tony Modica. A visual editor for reconfigurable object nets based on the eclipse graphical editor framework. *Arbeitsberichte aus dem Arbeitsberichte aus dem Fachbereich Informatik*, 2, 2007.

[260] Claudia Ermel, Sarkaft Shareef, and Winzent Fischer. Rons revisited: General approach to model reconfigurable object nets based on algebraic high-level nets. *Electronic Communications of the EASST*, 40, 2011.

[261] Radja Hamli, Allaoua Chaoui, Raida Elmansouri, and Ali Khebizi. Comprehensive framework-based reconfigurable object nets for managing dynamic protocols evolution. *International Journal of Organizational and Collective Intelligence (IJOCI)*, 13(1):1–33, 2023.

[262] Julia Padberg, Marvin Ede, Gerhard Oelker, and Kathrin Hoffmann. Reconnet: a tool for modeling and simulating with reconfigurable place/transition nets. *Electronic Communications of the EASST*, 54, 2012.

[263] Manel Houimli, Laid Kahloul, and Mohamed Khalgui. Multi-objective optimization and formal specification of reconfigurable manufacturing system using adaptive nsga-ii. In *2017 First International Conference on Embedded & Distributed Systems (EDiS)*, pages 1–6. IEEE, 2017.

[264] Genichi Taguchi, Subir Chowdhury, Yuin Wu, et al. *Taguchi's quality engineering handbook*, volume 1736. John Wiley & Sons Hoboken, NJ, 2005.

[265] Nery Riquelme, Christian Von Lücken, and Benjamin Baran. Performance metrics in multi-objective optimization. In *2015 Latin American Computing Conference (CLEI)*, pages 1–11. IEEE, 2015.

[266] H-E Tseng. Guided genetic algorithms for solving a larger constraint assembly problem. *International Journal of Production Research*, 44(3):601–625, 2006.

[267] Javad Sadeghi and Seyed Taghi Akhavan Niaki. Two parameter tuned multi-objective evolutionary algorithms for a bi-objective vendor managed inventory model with trapezoidal fuzzy demand. *Applied Soft Computing*, 30:567–576, 2015.

[268] Gonzalo Mejía, Juan Pablo Caballero-Villalobos, and Carlos Montoya. Petri nets and deadlock-free scheduling of open shop manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(6):1017–1028, 2017.

[269] Sarah Nazari, Behrang Sajadi, and Iman Sheikhansari. Optimisation of commercial buildings envelope to reduce energy consumption and improve indoor environmental quality (ieq) using nsga-ii algorithm. *International Journal of Ambient Energy*, 44(1):918–928, 2023.

[270] Bruno Ferreira, André Antunes, Nelson Carriço, and Dídia Covas. Nsga-ii parameterization for the optimal pressure sensor location in water distribution networks. *Urban Water Journal*, 20(6):738–750, 2023.

[271] Mostafa Akbari and Hossein Rahimi Asiabaraki. Modeling and optimization of tool parameters in friction stir lap joining of aluminum using rsm and nsga ii. *Welding International*, 37(1):21–33, 2023.

[272] Eduardo H Haro, Omar Avalos, Jorge Gálvez, and Octavio Camarena. An integrated process planning and scheduling problem solved from an adaptive multi-objective perspective. *Journal of Manufacturing Systems*, 75:1–23, 2024.

[273] Shin-ichi Inage. Proposal of the "total error minimization method" for robust design. *Engineering Science and Technology, an International Journal*, 22(2):656–666, 2019.

[274] Audrey Cerqueus and Xavier Delorme. Evaluating the scalability of reconfigurable manufacturing systems at the design phase. *International Journal of Production Research*, 61(23):8080–8093, 2023.

[275] Fu-Shiung Hsieh. An efficient method to assess resilience and robustness properties of a class of cyber-physical production systems. *Symmetry*, 14(11):2327, 2022.