SimpleSim: An Open-Source Python Simulator for SDM-EON Resource Allocation

Ramon A. Oliveira[1], **Helder Oliveira**[2]

[1]Universidade Federal do Pará (UFPA),Pará, Brazil
[2]Universidade de São Paulo (USP), São Paulo, Brazil

ramon.oliveira@itec.ufpa.br, helderoliveira@ime.usp.br

Abstract. *Space-Division Multiplexing Elastic Optical Networks (SDM-EONs) increase transmission capacity while significantly raising the complexity of the Routing, Modulation, Spectrum, and Space Allocation (RMSSA) problem set, demanding flexible simulation environments for evaluation. This paper presents SimpleSim, an open-source Python-based discrete-event simulator for prototyping and benchmarking RMSSA strategies in SDM-EONs. The tool adopts a modular architecture with external XML-based topology definition, integrates widely used scientific libraries, and provides automated statistical analysis with confidence intervals and built-in plotting capabilities. SimpleSim supports multi-core fiber modeling, crosstalk-aware allocation, and comparative evaluation of resource allocation algorithms under configurable traffic loads.*

1. Introduction

The continuous growth of data traffic has increased the demand for scalable and flexible optical network infrastructures. Elastic Optical Networks (EONs) enable fine-grained spectrum allocation [López et al. 2016], while Space-Division Multiplexing (SDM) expands transmission capacity through multi-core fibers. The integration of these paradigms results in Space-Division Multiplexing Elastic Optical Networks (SDM-EONs), which combine spectral flexibility with spatial parallelism.

Despite their capacity advantages, SDM-EONs significantly increase the complexity of resource allocation. The joint selection of routing paths, modulation formats, spectrum slots, and spatial cores, known as the Routing, Modulation, Spectrum, and Space Allocation (RMSSA) problem [Dixit et al. 2021], must consider constraints such as spectrum continuity, contiguity, and inter-core crosstalk. Due to the strong interdependence among these dimensions, analytical modeling becomes impractical, making simulation-based evaluation essential.

Modern SDM-EON research requires simulation platforms that not only support accurate modeling but also promote reproducibility and integration with data-driven workflows. Externalized topology definitions, standardized statistical reporting with confidence intervals, and compatibility with Python-based machine learning ecosystems are increasingly important for benchmarking heuristic and learning-based RMSSA strategies.

To address these requirements, this paper presents *SimpleSim*, a fully open-source Python-based discrete-event simulator designed for rapid prototyping and reproducible benchmarking in SDM-EON environments. We chose Python for its extremely diverse

ecosystem with several network, machine learning, and data analysis libraries that are actively maintained and widely used in academic research[Raschka et al. 2020]. SimpleSim combines modular architecture, XML-based external topology configuration, automated 95% confidence interval computation, and built-in visualization within a lightweight and extensible framework.

The main contributions of this work are:

- A modular SDM-EON simulation architecture separating topology management, event processing, allocation logic, and statistical analysis;
- Native support for multi-core fiber modeling with optional crosstalk-aware allocation;
- External XML-based topology configuration enabling reproducible experimentation;
- Automated generation of performance metrics with 95% confidence intervals and integrated plotting utilities;
- Seamless compatibility with Python-based scientific and machine learning libraries.

The remainder of this paper is structured as follows. Section 2 discusses existing simulation tools for optical networks and positions SimpleSim within this landscape. Section 3 details the architecture and main components of the simulator. Section 4 outlines the experimental workflow and reproducibility features. Section 5 presents illustrative benchmarking results generated by the tool. Finally, Section 6 summarizes the contributions and discusses future directions.

2. Related Simulation Tools

Simulation tools are essential for evaluating resource allocation strategies in optical networks. Although several platforms have been developed for WDM, EON, and SDM-EON research, important trade-offs remain regarding extensibility, reproducibility, and integration with modern scientific workflows.

FlexGridSim [Moura and Drummond] is a Java-based discrete-event simulator that supports EON and SDM-EON scenarios with multiple multi-core fiber configurations. It has been widely adopted for provisioning and routing studies. However, it does not natively provide automated statistical processing with confidence intervals or integrated plotting capabilities, requiring external post-processing. Moreover, its Java-based ecosystem may limit seamless interoperability with contemporary Python-based data analysis and machine learning libraries. The ONS simulator [R. Costa et al. 2016] is another Java-based platform designed to evaluate RWA and RMLSA strategies in WDM and EON networks. While it offers a structured simulation framework, it does not natively incorporate multi-core fiber modeling or inter-core crosstalk evaluation, restricting its direct applicability to SDM-EON studies.

Júnior *et al.* [Júnior et al. 2022] proposed a Python-based SDM-EON simulator that includes physical impairment modeling such as noise and crosstalk. Although it advances Python-native SDM experimentation, topology definitions are embedded directly in the source code, reducing flexibility for rapid experimentation across multiple network scenarios and complicating reproducibility. Beyond modeling features, reproducibility

has emerged as a critical issue in network simulation research. Hardcoded parameters, undocumented experimental configurations, and the absence of standardized statistical reporting often hinder fair comparison and replication of published results. In addition, the growing adoption of machine learning (ML) techniques for RMSSA optimization demands simulation environments that integrate naturally with Python-based data-driven workflows.

Table 1 summarizes key characteristics of representative SDM-EON simulation tools.

Table 1. Comparison of SDM-EON Simulation Tools

Tool	Language	SDM	Crosstalk	External Topology	Built-in Plotting	CI Support
FlexGridSim	Java	Yes	Partial	No	No	No
ONS	Java	No	No	Yes	No	No
Junior et al.	Python	Yes	Yes	No	Limited	No
SimpleSim	Python	Yes	Yes	Yes	Yes	Yes

As shown in Table 1, existing tools provide valuable simulation environments but present limitations in terms of external topology configuration, automated confidence interval computation, and built-in visualization. Java-based platforms such as FlexGridSim and ONS offer mature discrete-event infrastructures but limited integration with modern ML ecosystems. The Python-based simulator by Júnior *et al.* advances SDM modeling but relies on hardcoded topology definitions and does not natively automate statistical confidence reporting.

In contrast, SimpleSim combines native SDM and crosstalk-aware modeling, XML-based external topology definition, automated generation of performance metrics with 95% confidence intervals, and integrated plotting utilities within a modular Python architecture. By explicitly addressing reproducibility and ML compatibility, SimpleSim provides a lightweight and extensible platform for benchmarking both heuristic and learning-based RMSSA strategies.

3. Simulation Environment

SimpleSim adopts a modular layered architecture designed to promote extensibility and reproducible experimentation. Instead of a monolithic pipeline, the simulator organizes the workflow into independent components responsible for configuration management, topology abstraction, event processing, resource allocation, and statistical analysis.

Figure 1 illustrates the interaction among these components. The modular structure enables new RMSSA strategies to be incorporated without modifying the core simulation engine.

The simulation process begins with external configuration files and topology descriptions that define the experimental scenario. The network is instantiated as a graph abstraction, after which the discrete-event engine generates traffic demands and triggers allocation decisions. Performance metrics are automatically aggregated and exported for analysis.

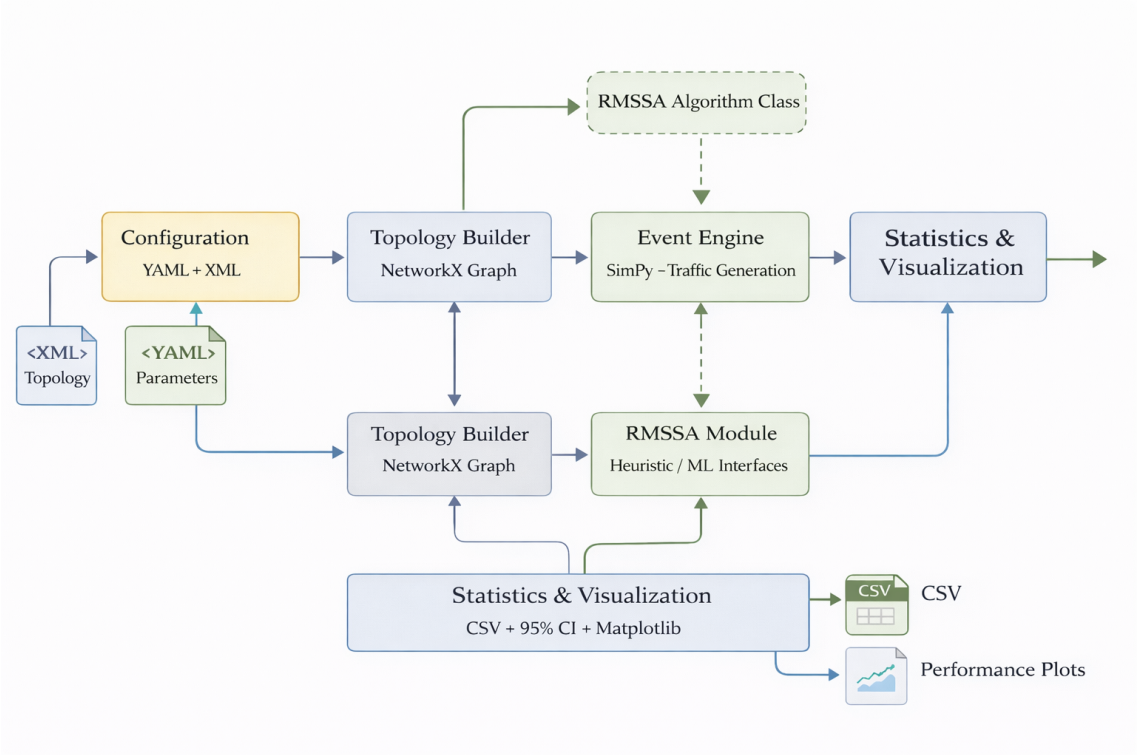


Figure 1. SimpleSim Simulation Architecture

3.1. Configuration Layer

SimpleSim utilizes customizable YAML configuration files for Simulation parameters setting and builds the network topology from topology XML files, separating experimental setup from implementation. These files specify fiber characteristics, traffic load intervals, simulation rounds, routing policies, crosstalk enforcement, and output structure. Therefore, the simulation parameters for any experiment can be carefully documented by producing a descriptive YAML configuration file particular to the experiment.

3.2. Topology Builder

Topologies defined in XML are parsed into directed graphs using NetworkX [Hagberg et al. 2008]. Nodes represent optical switches, while edges store link attributes such as length, number of cores, and available spectrum slots. This abstraction enables routing and crosstalk-aware allocation without modifying the simulator core.

3.3. Event Engine

The event engine is implemented using SimPy [SimPy 2024]. It manages traffic generation, request scheduling, and resource release events, maintaining the state of spectrum occupancy across cores and links. Allocation decisions are triggered upon each incoming request.

3.4. RMSSA Module Interface

The RMSSA module performs routing, core selection, and contiguous spectrum assignment. For each request, it receives the selected path, spectrum occupancy state, and traffic

parameters, returning either a feasible core-slot allocation or a blocking decision.

The interface is lightweight and decoupled from the event engine, allowing heuristic, metaheuristic, or machine learning-based strategies to be implemented as independent classes without altering the remaining simulation components.

3.5. Statistics and Visualization

During execution, metrics such as bandwidth blocking ratio (BBR), fragmentation, and inter-core crosstalk (CpS) are collected across multiple rounds. For each load level, SimpleSim computes average values and 95% confidence intervals, exporting structured CSV files. Built-in Matplotlib-based [Hunter 2007] utilities generate comparative performance plots.

4. Experimental Workflow and Demonstration

To demonstrate the benchmarking capabilities of SimpleSim, we present a representative experimental workflow commonly adopted in SDM-EON research. The objective is not merely to illustrate tool execution, but to highlight its support for reproducible experimentation, comparative evaluation, and rapid integration of allocation strategies.

4.1. Experimental Setup

All experiments are configured through external YAML parameter files and XML-based topology descriptions. This separation ensures that experimental scenarios can be replicated or modified without altering the simulator source code.

For demonstration purposes, we adopt a canonical SDM-EON configuration widely used in the literature: a seven-core multi-core fiber (MCF) with 320 spectrum slots per core and 12.5 GHz slot granularity, deployed over the NSF topology. Traffic load is varied across a predefined range of Erlang values, and for each load level multiple independent simulation rounds are executed to compute statistically significant results.

The configuration file specifies fiber characteristics, traffic load intervals, number of call attempts per round, routing policy, crosstalk enforcement, and the RMSSA strategy to be evaluated. This design allows systematic benchmarking under controlled and reproducible conditions.

4.2. Benchmarking of Allocation Strategies

To illustrate comparative evaluation, two classical allocation strategies are implemented: First-Fit (FF) and Best-Fit (BF). These strategies serve as reference heuristics for assessing bandwidth blocking ratio (BBR), fragmentation, and inter-core crosstalk (CpS).

For each offered load, SimpleSim automatically aggregates results across simulation rounds and computes average performance metrics along with 95% confidence intervals. The structured CSV output enables transparent reporting and facilitates external post-processing when necessary.

Although direct numerical comparison across different simulators is inherently challenging due to differences in traffic models and architectural assumptions, the observed trends in performance metrics are consistent with established behavior in SDM-EON literature, reinforcing the validity of the simulation environment.

4.3. Extensibility of the RMSSA Interface

SimpleSim is designed to simplify the integration of new RMSSA strategies. Implementing a new allocation approach requires defining a class that adheres to the standardized input–output interface described in Section 3. The allocation module receives the current network state and returns either a feasible core-slot assignment or a blocking decision.

This modular structure enables rapid experimentation with heuristic, metaheuristic, or learning-based strategies. In particular, reinforcement learning agents or external optimization routines can be incorporated into the allocation layer without modifications to the event engine or topology modules.

4.4. Reproducibility and Output Management

Simulation outputs are automatically stored in structured CSV files organized by experiment and allocation strategy. For each load level, SimpleSim exports performance averages and corresponding confidence intervals, ensuring transparency in statistical reporting. Additionally, built-in visualization utilities generate comparative performance curves directly from the generated data files. If multiple strategies are evaluated within the same output directory, the simulator automatically produces combined plots for side-by-side analysis. By externalizing configuration parameters, standardizing output formats, and automating statistical aggregation, SimpleSim supports reproducible benchmarking workflows aligned with contemporary research best practices.

SimpleSim is fully open-source and publicly available at: `https://github.com/Ramonnnnnn/SimpleSim`. The project is released under the MIT License. The repository contains installation instructions, execution guidelines, example configurations, and all necessary files to reproduce the experimental results presented in this work. Detailed documentation, including installation, execution, and step-by-step reproduction guidelines, is available at: `https://github.com/Ramonnnnnn/SimpleSim/tree/master/docs`.

5. Example Benchmark Results

This section presents representative results obtained with SimpleSim to demonstrate its benchmarking and statistical reporting capabilities. The objective is to illustrate the simulator workflow and automated generation of performance curves with confidence intervals, rather than to provide an exhaustive performance study.

5.1. Evaluation Scenario

Experiments were conducted using the NSF topology with a seven-core multi-core fiber configuration and 320 spectrum slots per core at 12.5 GHz granularity. Traffic load was varied between 500 and 1000 erlangs.

For each load level, multiple independent simulation rounds were executed. SimpleSim automatically aggregates the collected data and computes mean values with 95% confidence intervals, ensuring statistically robust results under stochastic traffic conditions. Two classical RMSSA heuristics were evaluated for demonstration purposes: First-Fit (FF) and Best-Fit (BF). These strategies were selected as baseline allocation approaches commonly adopted in optical network studies.

5.2. Performance Metrics

SimpleSim provides a set of core metrics frequently used in SDM-EON evaluation. In this example, we focus on three primary indicators:

Bandwidth Blocking Ratio (BBR), defined as the ratio between the total blocked bandwidth and the total requested bandwidth, reflecting the efficiency of the allocation strategy under varying traffic loads; Bandwidth Blocking Count Ratio (BCR), defined as the ratio between the number of blocked connection requests and the total number of connection requests, capturing request-level blocking behavior; Fragmentation, which quantifies the dispersion of available spectrum slots across cores and links, indicating how allocation decisions impact spectrum organization and future request accommodation; and Inter-core Crosstalk (CpS), which measures the average spatial interference per allocated slot, reflecting physical-layer effects inherent to multi-core fiber utilization.

In addition to these primary metrics, the simulator supports the extraction of complementary indicators such as connection blocking probability, spectrum utilization per core, and aggregate resource occupancy. Owing to the modular statistics layer, new metrics can be incorporated with minimal implementation effort.

5.3. Illustrative Results

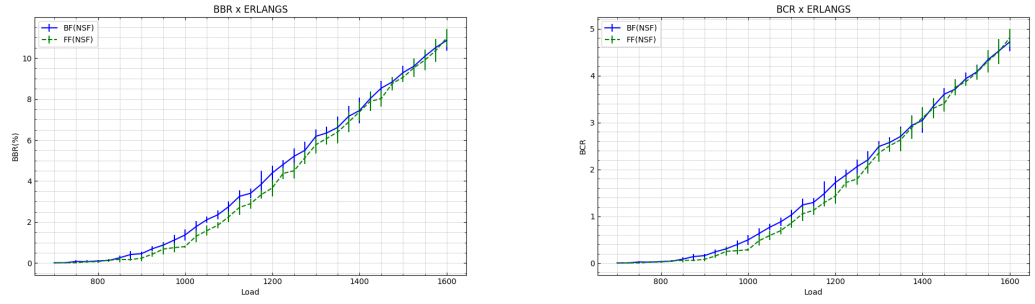
Figure 2 presents representative performance curves generated automatically by SimpleSim for the evaluated strategies. The plots include bandwidth blocking ratio (BBR), bandwidth blocking count ratio (BCR), fragmentation, and inter-core crosstalk (CpS), all reported with 95% confidence intervals. As expected, both BBR and BCR increase with traffic load due to progressive resource saturation. Differences between FF and BF reflect their distinct spectrum allocation behavior, particularly in terms of fragmentation patterns and spectrum compactness.

Fragmentation trends highlight the impact of allocation policy on spectrum organization, while CpS increases with load as spatial occupancy across cores grows. The automatic computation of confidence intervals illustrates the simulator's built-in statistical aggregation capabilities and supports transparent comparative benchmarking. These results demonstrate how SimpleSim enables reproducible evaluation of RMSSA strategies while automating data collection, statistical processing, and visualization.

6. Final Remarks

This paper presented SimpleSim, an open-source Python-based discrete-event simulator designed for prototyping and benchmarking RMSSA strategies in SDM-EON environments. The tool was conceived to address practical limitations observed in existing platforms, particularly regarding extensibility, reproducibility, and integration with modern data-driven research workflows.

SimpleSim combines modular architecture, XML-based external topology definition, automated 95% confidence interval computation, and built-in visualization capabilities within a lightweight framework. By separating configuration, topology abstraction, event management, allocation logic, and statistical processing, the simulator enables the rapid incorporation of new heuristic, metaheuristic, or machine learning-based strategies without modifying the simulation core.



(a) BBR (b) BCR

(c) Fragmentation (d) CpS

Figure 2. NSF Statistics for the Tested Algorithms

A central contribution of this work is the explicit emphasis on reproducible experimentation. Externalized configuration files, structured CSV outputs, and automated statistical aggregation facilitate transparent benchmarking and fair comparison of allocation approaches. Additionally, the Python-based implementation ensures seamless interoperability with scientific computing and machine learning ecosystems, lowering the barrier for ML-driven RMSSA research.

The complete source code, documentation, and reproducibility artifacts are publicly available, enabling immediate use and extension by the research community. By prioritizing modularity, openness, and reproducibility, SimpleSim aims to provide a robust experimental foundation for future SDM-EON resource allocation studies. Future developments include enhanced physical-layer modeling, extended quality-of-transmission (QoT) support, and deeper integration with learning-based optimization frameworks.

7. Demonstration Requirements and Planning

The demonstration of SimpleSim will be performed using a standard notebook computer with Python 3.10 or higher installed. The simulator is platform-independent and runs on Linux, Windows, and macOS systems under both ARM and x86 architectures. No specialized hardware or GPU resources are required. The demonstration setup requires only a notebook computer and a projector or HDMI-compatible display. Internet access is optional, as all necessary files, configurations, and benchmark scenarios are locally available.

During the session, we will present the simulator architecture, execute a representative RMSSA benchmark scenario, and demonstrate the automatic generation of CSV

outputs with 95% confidence intervals and comparative performance plots. The complete source code, documentation, and configuration files required to reproduce the experimental results presented in this paper are publicly available in the project repository.

References

Dixit, S., Batham, D., and Narwaria, R. P. (2021). Elastic optical network: A promising solution for future communication networks considering differentiated cos. In *Machine Intelligence and Smart Systems: Proceedings of MISS 2020*, pages 49–60. Springer.

Hagberg, A., Swart, P., and S Chult, D. (2008). Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Júnior, G. B. G., Pereira, H. A., and Félix, R. A. A. R. (2022). *Simulador de código aberto para redes ópticas considerando multiplexação por divisão espacial e o impacto de penalidades físicas*. Atena Editora.

López, V., Velasco, L., et al. (2016). Elastic optical networks. *Architectures, Technologies, and Control, Switzerland: Springer Int. Publishing*.

Moura, P. M. and Drummond, A. C. FlexGridSim: Flexible Grid Optical Network Simulator. http://www.lrc.ic.unicamp.br/FlexGridSim/.

R. Costa, L., de Sousa, L., Oliveira, F., Silva, K., Júnior, P., and Drummond, A. (2016). Ons: Simulador de eventos discretos para redes Ópticas wdm / eon.

Raschka, S., Patterson, J., and Nolet, C. (2020). Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, 11(4):193.

SimPy, T. (2024). Simpy. `https://simpy.readthedocs.io/en/latest/`.

Acknowledgement

Research partially funded by CNPq (grants no. #305489/2023-2) and by the São Paulo Research Foundation (FAPESP), Brasil. Process 2025/03079-4.