Universidade Federal de São Carlos – UFSCar
Centro de Ciências Exatas e de Tecnologia – CCET
Departamento de Computação – DC
Programa de Pós-Graduação em Ciência da Computação – PPGCC

Paulo Cesar Donizeti Paris

Simulador de Alto Nível para *Network-on-Chip*: Método para a Exploração de Estratégias de Mapeamento e Roteamento

São Carlos
Maio de 2024

Paulo Cesar Donizeti Paris

Simulador de Alto Nível para *Network-on-Chip*: Método para a Exploração de Estratégias de Mapeamento e Roteamento

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos , como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Arquitetura de Computadores

Orientador: Prof. Dr. Emerson Carlos Pedrino

São Carlos
Maio de 2024

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Defesa de Tese de Doutorado do candidato Paulo Cesar Donizeti Paris, realizada em 14/05/2024.

Comissão Julgadora:

Prof. Dr. Emerson Carlos Pedrino (UFSCar)

Prof. Dr. Helio Crestana Guardia (UFSCar)

Prof. Dr. Fredy João Valente (UFSCar)

Prof. Dr. José Hiroki Saito (UNIFACCAMP)

Prof. Dr. Mário Luiz Tronco (USP)

O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Ciência da Computação.

Dedico este trabalho a todos que sempre estiveram ao meu lado, especialmente à minha amada esposa, Carolina, que, com a bênção de Deus, presenteou-me com nosso tesouro mais precioso: nossa filha, Laura, a força que me impulsiona a cada amanhecer. Dedico também ao meu anjo da guarda, minha mãe, Zulmira (in memoriam), ao meu herói, meu pai, Sebastião, e à minha iluminada sobrinha, Daniele (in memoriam). Expresso minha profunda gratidão ao meu amigo de longa data e orientador, Prof. Dr. Emerson Carlos Pedrino, responsável por proporcionar a realização deste sonho, e ao eterno professor Hiroshi Tejima (in memoriam), o ser humano mais admirável e extraordinário que tive a honra de conhecer e com quem convivi.

Agradecimentos

Primeiramente, expresso minha eterna gratidão a Deus pelo dom da vida e à intercessão de Nossa Senhora Aparecida em minhas orações. Um agradecimento especial à minha esposa, Carolina, e à minha filha, Laura, pelo amor incondicional e por abdicarem de muitos momentos especiais para me acompanharem nesta longa jornada. Também sou grato pelo apoio de meus familiares: meu pai, Sebastião; sua esposa, Madalena; minhas irmãs, Luciene e Eva; meus sobrinhos, Arthur e Evandro; minhas sobrinhas, Isabelle, Lívia e Priscila; meus sogros, Regina e Heleno; e meus cunhados, Donizete e Alexandre. Valorizo imensamente os momentos de descontração proporcionados pelos meus queridos amigos, Gustavo, Alexandra e a filha deles, Helena.

Reconheço a contribuição indispensável do Prof. Dr. Emerson Carlos Pedrino, cuja dedicação e sabedoria foram fundamentais para manter minha resiliência e foco. Agradeço aos colegas do Departamento de Computação da UFSCar, incluindo alunos da graduação e da pós-graduação, professores como Dra. Marcela Xavier Ribeiro, Dra. Vânia Paula de Almeida Neris, Dra. Maria do Carmo Nicoletti, Dr. Márcio Merino Fernandes, Dr. Valter Vieira de Camargo, e aos técnicos administrativos Ivan, Darli, Vera, William, Aline, Nicanor e Dalila.

Agradeço, ainda, aos membros da banca, pela aceitação do convite. Por fim, estendo meu agradecimento ao Programa de Pós-Graduação em Ciência da Computação (PPGCC), à CAPES e à instituição UFSCar pelo suporte e oportunidades proporcionadas.

"Seja a mudança que você quer ver no mundo."
Mahatma Gandhi.

Resumo

Arquiteturas *many-core* podem conter dezenas ou centenas de núcleos de processamento em um único *chip*, sendo usadas em aplicações de alto desempenho devido à sua capacidade de execução paralela. Embora ofereçam grande potencial para explorar paralelismo em nível de tarefas, enfrentam desafios, particularmente na divisão e comunicação eficiente entre elas. A comunicação entre núcleos é essencial, especialmente com a utilização da *Network-on-Chip* (NoC), que busca resolver problemas de escalabilidade e eficiência energética. Isso abre caminho para a otimização dessas abordagens, tanto em termos de suporte de *hardware* quanto de estratégias de mapeamento e roteamento. Diversos simuladores específicos para NoCs foram desenvolvidos para avaliar e otimizar esse potencial. No entanto, em sua maioria, são complexos, trabalhando em níveis baixos da arquitetura, como nas características funcionais e de desempenho dos roteadores, e não abrangem estratégias de mapeamento. Embora sejam excelentes ferramentas especializadas, oferecem pouca flexibilidade, ajustando-se a cenários específicos ou configurações fixas. Isso pode gerar uma demanda por estudos comparativos mais simples, com uma curva de aprendizado mais baixa e maior flexibilidade. Para complementar esses esforços e preencher possíveis lacunas, esta tese propõe um método para explorar estratégias de mapeamento e roteamento em alto nível, denominado Simulador NoC. Tal abordagem possibilita estudos iniciais e comparações de projetos de *many-core*, incluindo algoritmos de otimização customizáveis para mapeamento de tarefas paralelas de complexidade arbitrária, assim como algoritmos de roteamento. Com sua abordagem simplificada e modular, o simulador proposto permite aos usuários distinguir cada processo da simulação, desde a configuração de parâmetros até a avaliação de métricas de desempenho, com foco no consumo de energia. Resultados experimentais confirmaram sua confiabilidade na estimativa do consumo de energia simplificado, demonstrando sua utilidade para análises de exploração de espaço de projeto e comparações entre diferentes estratégias de mapeamento e roteamento. Constitui-se, assim, em uma abordagem interessante para atividades de ensino e pesquisa na área.

Palavras-chave: *many-core*, *network-on-chip*, mapeamento de aplicações, algoritmos de mapeamento, algoritmos de roteamento, simuladores para NoC, métrica de consumo energético.

Abstract

Many-core architectures can contain dozens or even hundreds of processing cores on a single chip, being used in high-performance applications due to their parallel execution capability. Although they offer great potential for task-level parallelism, they face challenges, particularly in the efficient division and communication among tasks. Communication among cores is essential, especially with the use of Network-on-Chip (NoC), which aims to solve scalability and energy efficiency issues. This paves the way for optimizing these approaches, both in terms of hardware support and mapping and routing strategies. Several specific simulators for NoCs have been developed to evaluate and optimize this potential. However, the majority are complex, operating at lower levels of the architecture, such as in the functional and performance characteristics of routers, and do not encompass mapping strategies. While they are excellent specialized tools, they offer little flexibility, being suited to specific scenarios or fixed configurations. This can create a demand for simpler comparative studies, with a lower learning curve and greater flexibility. To complement these efforts and fill possible gaps, this thesis proposes a method for exploring high-level mapping and routing strategies, called the NoC Simulator. This approach enables initial studies and comparisons of many-core designs, including customizable optimization algorithms for mapping parallel tasks of arbitrary complexity, as well as routing algorithms. With its simplified and modular approach, the proposed simulator allows users to distinguish each process of the simulation, from parameter configuration to performance metric evaluation, focusing on energy consumption. Experimental results confirmed its reliability in estimating simplified energy consumption, demonstrating its utility for design space exploration analyses and comparisons among different mapping and routing strategies. Thus, it constitutes an interesting approach for teaching and research activities in the field.

Keywords: many-core, network-on-chip, application mapping, mapping algorithms, routing algorithms, NoC simulators, energy consumption metrics.

Lista de ilustrações

Lista de tabelas

Lista de siglas

ACA	Andean Condor Algorithm
ACO	Ant Colony Optimization Algorithms
BB	Branch and Bound
BPS	Bits per second
CMOS	Complementary Metal-Oxide-Semiconductor
CUDA	Compute Unified Device Architecture
DAG	Directed Acyclic Graph
DSE	Design Space Exploration
DSP	Digital Signal Processor
Ebit	Bit Energy
ER	Encircle Routing
FIFO	First In, First Out
Flits	Flow Control Digits
GA	Genetic Algorithms

GPGPU	General-Purpose Computing on Graphics Processing Units
GPU	Graphics Processing Units
HES	Hybrid Exact & Search
ILP	Integer Linear Programming
IP	Intellectual Property
KTH	Royal Institute of Technology
LSC	Laboratório de Sistemas Computacionais
MM	Mathematical Morphology
NF	Negative-First
NI	Network Interface
NL	North-Last
NMAP	Near-optimal Mapping
NoC	Network-on-Chip
OE	Odd-Even
PE	Processing Element
PlatEMO	Evolutionary Multi-objective Optimization Platform
PSO	Particle Swarm Optimization
SA	Simulated Annealing
SIMD	Single Instruction, Multiple Data
SoC	System-on-Chip
TS	Tabu Search
VLIW	Very Long Instruction Word
WF	West-First

Sumário

Capítulo 1

Introdução

Neste capítulo, é apresentada uma visão geral do trabalho desenvolvido, incluindo o seu contexto científico, problemas de interesse que motivaram o desenvolvimento das soluções propostas e objetivos que guiaram o seu desenvolvimento.

1.1 Contexto

Nas últimas décadas, tem-se observado grande avanço no projeto e implementação de sistemas computacionais, os quais apresentam desempenho continuamente crescente, bem como aumento de capacidade de memória e armazenamento e, mais recentemente, eficiência energética, quando comparados a sistemas de gerações anteriores.

Em termos de *hardware*, os avanços foram viabilizados pela confirmação da Lei de Moore, que trata da evolução no projeto e implementação de dispositivos semicondutores (MOORE, 1998). Isso também impulsionou desenvolvimentos no *software*, incluindo sistemas operacionais, compiladores e *frameworks* de programação. Mais recentemente, a combinação desses recursos possibilitou a disseminação de sistemas baseados em tecnologias avançadas e complexas, como *Data Science* e *Artificial Intelligence* (AINSWORTH; JONES, 2019; MONDAL, 2020; MATHEW;

AMUDHA; SIVAKUMARI, 2021).

O quase esgotamento das possibilidades de aumento de desempenho de microprocessadores com um único núcleo, observado no início dos anos 2000, deu origem às chamadas arquiteturas *multi-core* (GEER, 2005). Tais chamadas apresentam grande potencial para a exploração de paralelismo na execução de tarefas, seja individualmente ou em conjunto. Esse modelo se mostrou eficiente, possibilitando a evolução desde microprocessadores simples, com dois ou mais núcleos, até arquiteturas de alto desempenho, com centenas de núcleos de processamento, os chamados *many-core* (VAJDA, 2011). Esses processadores podem ser encontrados em diversas configurações, seja em processadores especializados para computação de alto desempenho, homogêneos ou heterogêneos quanto ao tipo de núcleo, e até mesmo em sistemas completos, customizados como *System-on-Chip* (SoCs) (TORRES et al., 2011).

Os microprocessadores escalares tradicionais são essencialmente arquiteturas orientadas para a latência, tendo como objetivo minimizar o tempo de execução de um único programa sequencial, evitando a latência no nível da tarefa sempre que possível. Os microprocessadores *multi-core* refletem uma tendência para *designs* um pouco menos agressivos, mas ainda focados em ganhos de latência. Por outro lado, os processadores orientados para *throughput* surgem da suposição de que serão apresentados a eles cargas de trabalho nas quais o paralelismo é abundante, como é o caso dos *many-core* (GARLAND; KIRK, 2010).

Processadores *many-core* se tornaram mais escaláveis do que os *multi-core*, pois não são limitados pela estrutura de barramentos na comunicação entre os núcleos de processamento (ABDALLAH, 2013). Nesse contexto, outra tecnologia de comunicação denominada *Network-on-Chip* (NoC) (BENINI; MICHELI, 2002; DALLY; TOWLES, 2001) tem se mostrado adequada para suportar esses sistemas, sendo capaz de lidar com um número crescente de núcleos de processamento sem sofrer os gargalos de desempenho típicos dos sistemas que dependem exclusivamente de barramentos (LIU; KATO; EDAHIRO, 2019). Uma implementação fundamental nas estruturas das NoCs é a topologia *Mesh* 2D, conhecida por sua eficácia na conexão dos *Processing Elements* (PEs), utilizando roteadores para interligar cada um deles a até quatro vizinhos, rotulados como Norte, Sul, Leste e Oeste (VILLAESCUSA; RIVAS; HARBOUR, 2023).

Entretanto, na literatura são identificados alguns desafios no uso de NoCs nesse nicho. Eles incluem estratégias de mapeamento de tarefas (representadas por um *Directed Acyclic Graph* (DAG)), focadas na otimização da rede, e de roteamento, que visam a eficiência da comunicação, podendo ser determinísticas ou adaptativas (REDDY; KAR, 2022; BHASKAR; VENKATESH, 2021). As NoCs são geralmente avaliadas por parâmetros de desempenho, como consumo de energia, área, latência, *throughput* e largura de banda de comunicação, entre outros (BHASKAR, 2021). Também se observa um interesse crescente no desenvolvimento de meta-heurísticas e outros algoritmos para esses propósitos (SHARMA et al., 2023a; SIKANDAR et al., 2021a).

Diante desse cenário, diversos simuladores foram construídos e adotados para essa avaliação, visando obter informações mais precisas sobre a dinâmica da rede como um todo e, assim, evitar problemas críticos na implementação final. Diversas abordagens para lidar com essa questão foram encontradas na literatura especializada, várias delas descritas em levantamentos conduzidos por Khan et al. (2017a) e Al-Hchaimi et al. (2021). Um exemplo é o simulador ORION, que foca no desempenho de componentes individuais, como potência e área do roteador. Por essa razão, ele é frequentemente incorporado a outros simuladores para avaliar essas métricas (GU, 2011).

Existem outros modelos para estimar o consumo de energia em redes do tipo NoC. Um deles, pertencente à biblioteca padrão CMOS, é utilizado pelo simulador NoCTweak. Já o modelo *Ebit* é adotado pelo simulador NoCmap. Outros simuladores, como Noxim, Nirgam, Nostrum e BookSim, são projetados para medir o desempenho da rede completa. Como as NoCs herdam várias características de redes de computadores gerais, esses simuladores também podem ser usados em simulações de sistemas NoC (AL-HCHAIMI et al., 2021; KHAN et al., 2017b).

Entretanto, após uma avaliação mais detalhada em nível de instalação, configuração e testes, constatou-se que os principais simuladores específicos para NoC identificados se concentram mais em aspectos de baixo nível da rede, com vários algoritmos de roteamento, mas oferecem poucas opções para a exploração de projetos de algoritmos para mapeamento de tarefas. Embora permitam que os usuários criem e incorporem seus próprios algoritmos e estratégias, fazê-lo não se mostrou prático ou simples, uma vez que exige conhecimentos aprofundados sobre

a arquitetura considerada pelo simulador.

Nesse contexto, identificou-se a necessidade de uma nova abordagem nos padrões de um simulador que seja mais simplificado e voltado para o entendimento conceitual, para a análise e melhoria do desempenho e da comunicação da rede. A complexidade inerente à customização de algoritmos para essas finalidades, aliada à diversidade de métricas de desempenho a serem consideradas, ressalta a importância de uma metodologia adequada e viável para lidar e compreender os desafios específicos enfrentados nas NoCs, criando assim um ambiente favorável para a exploração de soluções capazes de entender a eficácia e superar as possíveis limitações.

1.2 Definição do Problema

Apesar de seu grande potencial de desempenho, os sistemas *many-core* enfrentam desafios significativos para serem explorados de maneira eficiente, especialmente no que se refere ao mapeamento de tarefas e à comunicação entre núcleos de processamento (SINGH et al., 2013). Esse problema, conhecido desde os primeiros computadores paralelos nos anos 70 e 80, continua presente devido à complexidade e à capacidade crescente das arquiteturas atuais (WEICHSLGARTNER et al., 2018). Diante desse contexto, torna-se necessária a adoção de estratégias de *Design Space Exploration* (DSE) (SINGH et al., 2021; CARDOSO; COUTINHO; DINIZ, 2017b).

Buscando viabilizar a utilização da DSE, é necessário um método que se adapte a ela, ou seja, um ambiente de simulação de alto nível que não exija uma longa curva de aprendizado. Esse sistema deve conter uma descrição abstrata com todas as características da estrutura NoC, mas sem detalhes técnicos do *hardware*, facilitando o desenvolvimento e a experimentação com novas topologias de rede e algoritmos para mapeamento e roteamento. Assim, busca-se avaliar e validar ideias iniciais de projeto, produzindo métricas de desempenho mais simples para análise e comparação, as quais devem guiar a tomada de decisões para a continuidade de um projeto baseado em sistemas *many-core*.

1.3 Hipótese para a Resolução do Problema

A partir do problema estabelecido na seção 1.2, adotou-se a seguinte hipótese para o desenvolvimento deste trabalho:

> *É possível desenvolver um método baseado nas características da estratégia DSE para NoCs, adotando um nível de abstração mais alto do que o dos simuladores existentes, buscando facilitar a exploração de um grande número de opções de algoritmos de mapeamento e roteamento aplicados a topologias de redes distintas. Isso inclui também o estudo conceitual da dinâmica geral da rede baseada em pacotes abstratos e dos perfis das aplicações, representadas como um DAG.*

Para validar a hipótese apresentada, foram adotados os seguintes critérios:

a) O simulador deverá produzir resultados que possam ser facilmente utilizados para gerar tabelas de comparação e gráficos, baseados na métrica de custo simplificado de energia;

b) Os resultados obtidos devem permitir uma compreensão clara do desempenho relativo entre duas ou mais opções de projeto, especialmente quando comparados com simulações de maior detalhamento e precisão, utilizando a métrica de custo simplificado de energia;

c) O simulador deverá permitir escalabilidade, permitindo a incorporação de novas opções de arquiteturas, aplicações e, principalmente, algoritmos de mapeamento e roteamento de maneira eficiente e sem complexidade excessiva.

1.4 Objetivos do Trabalho

1.4.1 Objetivo Geral

O trabalho desenvolvido teve como objetivo geral conceber e desenvolver um simulador de alto nível para arquiteturas *many-core* baseadas em NoCs, organizado em módulos. Esse simulador deve estimar parâmetros de desempenho desses

sistemas sem se ater a detalhes específicos de *hardware*. Visa ser mais adequado para estudos e comparações iniciais, incluindo algoritmos de mapeamento e roteamento, no contexto de DSE. O objetivo foi oferecer características que permitam aos usuários assimilar os conceitos por meio de estudos em um ambiente controlado e acessível.

1.4.2 Objetivos Específicos

Para atingir o objetivo geral proposto, foram definidos os seguintes objetivos específicos:

1. Conceber e desenvolver um simulador para auxílio à atividade de DSE, denominado `Simulador NoC`, o qual engloba em seu escopo dois blocos principais: a representação da arquitetura NoC e um repositório de algoritmos para mapeamento e roteamento.

2. Conceber e desenvolver os seguintes submódulos que irão integrar o *Simulador NoC*, os quais têm suas funcionalidades específicas implementadas e executadas através de funções de programação:

    a) Módulo de configuração do ambiente e dos parâmetros da arquitetura *many-core*, representada como uma NoC, incluindo: número de linhas, número de colunas, dimensão do *grid*, tipo de topologia, número de soluções, número de evoluções e número de execuções;

    b) Módulo de conversão das bases de dados de entrada para o formato de uma matriz utilizada pelo simulador proposto;

    c) Módulo de geração de pacotes abstratos, que segue a estrutura do arquivo gerado pelo módulo anterior, representando o padrão dos DAGs de complexidade arbitrária;

    d) Módulo da estrutura do roteador, com a tabela e os campos necessários para abstração da NoC;

    e)

    f) Módulo para geração e configuração das métricas, com foco na métrica de custo simplificado de energia neste trabalho;

g) Módulo para geração e consolidação dos resultados das simulações, baseado nas métricas.

3. Desenvolver um ambiente de simulação para o usuário que minimize a necessidade de conhecimentos avançados sobre o simulador e detalhes de simulação não relevantes para os objetivos propostos;

4. Integrar o simulador com a plataforma PlatEMO (TIAN et al., 2017), possibilitando assim grande expansão na base de algoritmos que podem ser customizados para estratégias de mapeamento;

5. Validar a estratégia proposta por meio de experimentos, usando o método desenvolvido, sendo consideradas inicialmente as seguintes análises:

   a) Avaliação do comportamento e eficácia dos algoritmos de mapeamento em NoCs, com foco na métrica de consumo simplificado de energia;

   b) Verificação da eficiência de diferentes algoritmos de roteamento em NoCs, utilizando como base um dado algoritmo de mapeamento de aplicações escolhido pelo usuário;

6. Avaliação comparativa da metodologia proposta e desenvolvida no que diz respeito à sua adequação como ambiente de simulação de apoio para análises do tipo DSE, e estudo da viabilidade de incorporação de possíveis aprimoramentos.

1.5 Materiais e Desenvolvimento

O desenvolvimento do projeto foi realizado no Laboratório de Sistemas Computacionais (LSC), vinculado ao Departamento de Computação da Universidade Federal de São Carlos (UFSCar). A plataforma *Periódicos Capes* foi o principal meio de acesso às referências bibliográficas que serviram de base para estudos e formulações teóricas do projeto. Em termos de *software*, os seguintes sistemas e ferramentas foram utilizados e instalados:

a) Matlab: Ambiente para desenvolvimento do simulador proposto (MATLAB, 2010);

b) PlatEMO: Plataforma para experimentação com algoritmos evolutivos;

c) TGFF: Ferramenta projetada para fornecer um meio flexível e padrão de gerar DAGs pseudoaleatórios para uso em pesquisas de escalonamento e alocação, representando *benchmarks* sintéticos (DICK; RHODES; WOLF, 1998);

d) *Benchmarks* de referência: DAGs representando aplicações de interesse para os objetivos do projeto, com diversificação no número de tarefas.

Em termos de *hardware*, o desenvolvimento foi realizado em equipamentos com a seguinte configuração, apresentada na Tabela 1:

Tabela 1 – Configuração dos Equipamentos Utilizados

Configurações dos Computadores		
Componente	**Especificação 1**	**Especificação 2**
CPU:	AMD Ryzen 7 5700G 3.8GHz	Intel Core i7 3.60GHz
Placa Gráfica:	Radeon Graphics integrada	NVIDIA GeForce RTX 3080
Memória RAM:	16GB	32GB
Sistema Operacional:	Windows 10 Pro, versão 21H2	Linux Ubuntu 18.04
Armazenamento:	SSD 500GB	SSD 500GB

1.6 Contribuições do Trabalho

Em termos científicos, a principal contribuição deste trabalho foi a proposição, concepção e validação de uma abordagem de alto nível de abstração para a simulação de NoCs. Ela permite o acesso a uma vasta gama de algoritmos evolutivos, facilitando a experimentação e análise inicial em estudos de exploração do espaço de soluções de projeto para aplicações baseadas em sistemas *many-core*. Demonstrou-se ser viável e vantajosa para estudos dessa natureza, quando comparada ao uso dos simuladores convencionais para NoCs relatados em publicações da área. Baseando-se em extensa busca na literatura especializada, acredita-se que esta é a primeira implementação de um simulador para NoCs que oferece a possibilidade de experimentação com um número significativamente maior de algoritmos que podem ser customizados para o mapeamento de tarefas, assim como

de roteamento, em comparação a outros simuladores estudados, elevando, assim, as possibilidades de experimentação e análise.

Em termos práticos, o trabalho resultou no desenvolvimento e disponibilização de um simulador de alto nível de abstração para NoCs. Este se caracteriza pela adoção de um conjunto de funções e *scripts*, além da simplificação da integração de um número considerável de algoritmos adaptáveis ao mapeamento de aplicações e à comparação de estratégias de roteamento, com destaque para aqueles disponibilizados pela plataforma PlatEMO. Tais características provaram ser úteis para agilizar a realização de experimentos e a análise de dados relacionados à execução de aplicações em arquiteturas *many-core* baseados em NoCs. Isso representa um avanço significativo em termos de usabilidade e eficiência em comparação a outros simuladores de NoCs citados e testados anteriormente à implementação do projeto.

Os experimentos conduzidos e relatados nesta tese levaram a algumas conclusões não facilmente explicitadas utilizando outras ferramentas, confirmando a capacidade do simulador em produzir dados que permitiram aos usuários avaliar, validar e comparar estratégias de mapeamento e de roteamento para aplicações em sistemas *many-core* baseados em NoCs.

Como contribuições acadêmicas, a metodologia disponibilizada é de código aberto e mostra-se extensível em termos de funcionalidades, podendo, assim, ser adotada como base para outros trabalhos de pesquisa na área. Além disso, devido ao seu alto nível de abstração, pode ser utilizada como ferramenta para atividades de ensino, abordando temas como processamento *many-core*, NoCs, mapeamento e de roteamento, e estudos comparativos de algoritmos evolutivos, entre outros.

1.6.1 Publicações do Autor

Por fim, registram-se os seguintes artigos científicos diretamente relacionados à tese de doutorado, descrita neste documento:

a) **IEEE SMC 2023 - PUBLICADO:** Paulo Cesar Donizeti Paris, Emerson Carlos Pedrino: Packet simulator tool for many-core systems. IEEE International Conference on Systems, Man, and Cybernetics (SMC 2023), p.56–61, 2023.

b) **ICAE 2024 - SUBMETIDO em 07/01/2024:** Paulo Cesar Donizeti Paris, Emerson Carlos Pedrino: A High-Level Simulator for Network-on-Chip. Integrated Computer-Aided Engineering (ICAE).

Adicionalmente, durante o período de doutoramento, contribuí em trabalhos que resultaram na publicação de outros dois artigos científicos (sendo um deles relacionado ao meu projeto de mestrado), os quais julgo terem contribuído para minha formação como pesquisador, um dos objetivos do curso de doutorado:

a) **Przeglad Elektrotechniczny 2021 - PUBLICADO:** Rafael Givanildo, Denis Lima; PARIS, Paulo Cesar Donizeti Paris, Emerson Carlos Pedrino: Low-Cost and Accuracy Smart Meter Prototype for Smart Grids. Przeglad Elektrotechniczny, v.97, p.72-77, 2021.

b) **Genetic Programming and Evolvable Machines 2017 - PUBLICADO:** Emerson Carlos Pedrino, PParis, Emerson Carlos Pedrino, Denis Lima, Valentin Obac Roda: Software review: CGP-Library. Genetic Programming and Evolvable Machines, v.18, p.1-4, 2017.

1.7 Organização do Texto

Este documento tem seu texto principal organizado em capítulos, cujos conteúdos são resumidos a seguir:

1. Introdução: oferece uma visão geral do projeto desenvolvido, incluindo seu contexto, problema de pesquisa, hipótese, objetivos, métodos e materiais e principais contribuições;

2. Fundamentação Teórica: realiza uma revisão *top-down* dos conceitos relacionados à área do projeto, abrangendo arquiteturas *many-core*, sistemas de interconexão, NoCs, estratégias de mapeamento de tarefas, roteamento de comunicações, parâmetros de desempenho, algoritmos evolutivos e exploração do espaço de soluções de projeto;

3. Trabalhos Relacionados: apresenta os principais trabalhos científicos relacionados aos objetivos do projeto, com ênfase em simuladores de NoCs;

4. Metodologia e Desenvolvimento: concepção e desenvolvimento do simulador: detalha o método adotado para a estruturação do simulador e sua implementação;

5. Resultados e Discussões: experimentos de validação do método e ferramenta implementada: apresenta resultados experimentais para a validação da hipótese estabelecida para o projeto, seguido por experimentos de aplicação da ferramenta em análise do tipo DSE, demonstrando resultados experimentais de usos típicos que se beneficiariam da ferramenta desenvolvida;

6. Conclusões: considerações finais sobre o desenvolvimento do trabalho, os resultados alcançados e possíveis desenvolvimentos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo, é descrita uma revisão de conceitos e fundamentação teórica que compõem o contexto e objetivos deste projeto. Serão tratados tópicos relacionados com arquiteturas, many-core, NoCs, algoritmos de mapeamento de tarefas e roteamento de pacotes, métricas de desempenho e exploração de espaço de soluções de projeto.

2.1 Arquiteturas *Many-core*

As arquiteturas *many-core*, caracterizadas pela integração de muitos núcleos em uma única unidade computacional, marcam uma evolução significativa na tecnologia de processamento. Destinadas a superar as limitações dos sistemas *multi-core* tradicionais, elas habilitam o processamento simultâneo de múltiplas tarefas, ampliando o potencial de desempenho e a eficiência energética em diversas aplicações (ZARRIN; AGUIAR; BARRACA, 2017). Esta transição oferece oportunidades para inovação em diversas áreas de aplicações, como visão computacional (*Computer Vision*) (LIU et al., 2022), ciência de dados (*Data Science*) (AINSWORTH; JONES, 2019) e aprendizado de máquina, usando redes neurais (*Deep Learning*) (MATHEW; AMUDHA; SIVAKUMARI, 2021), entre outras.

Sua capacidade de processar segmentos de dados de forma independente faci-

lita experiências de usuário aprimoradas, com interfaces intuitivas, análises aceleradas e respostas imediatas a comandos complexos. A maximização dos benefícios proporcionados por essas arquiteturas requer que o desenvolvimento de *software* acompanhe tais inovações. Isso envolve adotar práticas de programação que distribuam tarefas eficazmente entre os núcleos, otimizando a operação das aplicações (ABUMWAIS, 2023). Particularmente vantajosas em computação de alto desempenho e simulações científicas, elas permitem que cada núcleo processe segmentos independentes, diminuindo o tempo total de cálculo.

Seguindo esses conceitos, várias áreas começaram a ser auxiliadas pelo potencial dessa abordagem, com projetos como o `Larrabee`, uma plataforma voltada para computação visual, utilizando múltiplos núcleos x86 aliados a uma unidade de processamento vetorial e blocos lógicos de função fixa. Ele foi projetado para fornecer alto desempenho em tarefas de computação gráfica e processamento paralelo, aproveitando a flexibilidade da arquitetura x86 para suportar uma ampla gama de aplicativos e cargas de trabalho. Além disso, a inclusão de unidades de processamento vetorial permite operações eficientes em grandes conjuntos de dados, enquanto os blocos lógicos de função fixa otimizam tarefas específicas, melhorando o desempenho geral do sistema. Essa combinação de recursos torna o projeto uma solução poderosa para ambientes de computação visual exigentes e aplicações de alto desempenho (SEILER et al., 2009).

Além da computação visual, o potencial das arquiteturas *many-core* se estende a aplicações de processamento de rede, em que técnicas como o agrupamento dinâmico de núcleos na arquitetura `Raw` demonstram melhorias significativas no desempenho de processadores *many-core*, especialmente em otimizar a distribuição de carga de trabalho entre os núcleos (KAZEMPOUR; FEDOROVA; ALAGHEBAND, 2008). Esta abordagem realça a adaptabilidade das arquiteturas *many-core* para atender a exigências específicas de eficiência e desempenho em comunicações de rede.

A evolução para sistemas heterogêneos é evidenciada pelo desenvolvimento de modelos de desempenho *cross-core*, que otimizam a alocação de recursos em arquiteturas *many-core* heterogêneas. Eles permitem uma execução mais eficiente de *threads*, maximizando a utilização dos núcleos mais adequados para cada tarefa. Esse conceito é fundamental para compreender como diferentes tipos de núcleos

podem coexistir e colaborar dentro de um sistema, visando à otimização global do desempenho. Esses modelos distribuem *threads* nos núcleos mais adequados, levando em consideração afinidade de cache, latência e *throughput*, melhorando significativamente o desempenho (PINHEIRO; ROMA; TOMÁS, 2017).

A investigação sobre algoritmos de ordenação paralela evidencia a importância das arquiteturas *many-core* na eficiência energética e velocidade de processamento de dados. Métodos desenvolvidos para *arrays* de processadores *many-core* de granularidade fina demonstram como tarefas computacionais intensivas podem ser otimizadas para alcançar desempenho superior com menor consumo de energia, ressaltando o papel crítico dessas arquiteturas em avanços tecnológicos em diversos domínios (STILLMAKER et al., 2020).

O desenvolvimento de *hardware* e *software* escaláveis representa um desafio significativo, que exige estratégias para melhorar o desempenho, reduzir o consumo energético e aumentar a resiliência. O planejamento cuidadoso da distribuição de tarefas, um modelo frequentemente adotado na literatura, assegura eficiência e confiabilidade operacional.

2.2 Network-on-Chip

Desde a publicação dos artigos (BENINI; MICHELI, 2002) e (DALLY; TOWLES, 2001), as pesquisas em NoCs têm sido fundamentais para a evolução da computação de alto desempenho. Elas se concentram em reduzir a latência e o consumo de energia, otimizando componentes como roteadores. Com os avanços nas tecnologias de semicondutores, essas arquiteturas se tornaram mais complexas, buscando se adaptar às novas demandas, como inteligência artificial. Tornaram-se, assim, essenciais para o desenvolvimento de arquiteturas *multi-core* e *many-core*. Essas inovações atendem às necessidades dos sistemas computacionais embarcados atuais, destacando sua importância no desenvolvimento de tecnologias integradas complexas (ZHANG et al., 2024).

2.2.1 Arquitetura NoC

A arquitetura de uma NoC é estruturada a partir de quatro componentes essenciais: *Processing Elements* (PEs) - Incluem processadores, DSPs, GPUs, aceleradores de *hardware* específicos e microcontroladores, desempenhando as funções de processamento de dados (YAN; MENG; CHEN, 2024); Roteadores - Implementam os protocolos de comunicação e direcionam os dados, gerenciando o tráfego, decidindo o caminho dos pacotes para alcançar seus destinos e garantindo a integridade e a ordem dos dados transmitidos. Utilizam técnicas avançadas de roteamento e controle de congestionamento para otimizar o desempenho (TSAI et al., 2012); *Network Interfaces* (NIs) - Estabelecem a conexão lógica entre os PEs e a rede, atuando como tradutores que adaptam os dados gerados pelos PEs ao formato de pacotes da NoC. Garantem comunicação eficiente e ordenada, lidando com enfileiramento, controle de fluxo e sincronização dos dados (PHING et al., 2019); e *Links* - Conectam fisicamente os nós computacionais (a combinação de um PE com um roteador) e seus vizinhos através de fios, suportando múltiplos canais físicos e lógicos, com a transmissão de dados assegurada por um protocolo de sincronização, podendo utilizar *FIFOs* (MISHRA; CHARLES, 2021).

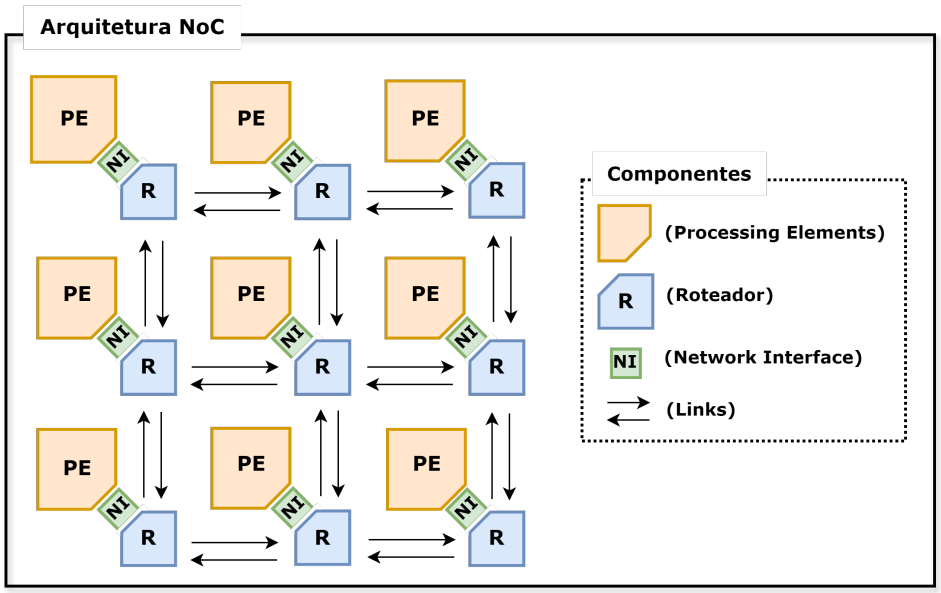

Figura 2.1 – Exemplo de uma Arquitetura Genérica NoC.
(Adaptado de (TSAI et al., 2012))

Nesse contexto, visualmente representado pela Figura 2.1, os elementos de processamento atuam como origem e destino para os pacotes, formando pares de comunicação nos quais ocorrem o tráfego e o processamento de dados inerentes à rede e ao problema em questão.

2.2.1.1 Roteador

A arquitetura do roteador NoC incorpora portas de entrada e saída direcionais, que incluem as direções cardeais, Norte (*North*), Sul (*South*), Leste (*East*) e Oeste (*West*), além de uma porta local para interface com o PE (*Processing Element*) (Figura 2.2a).

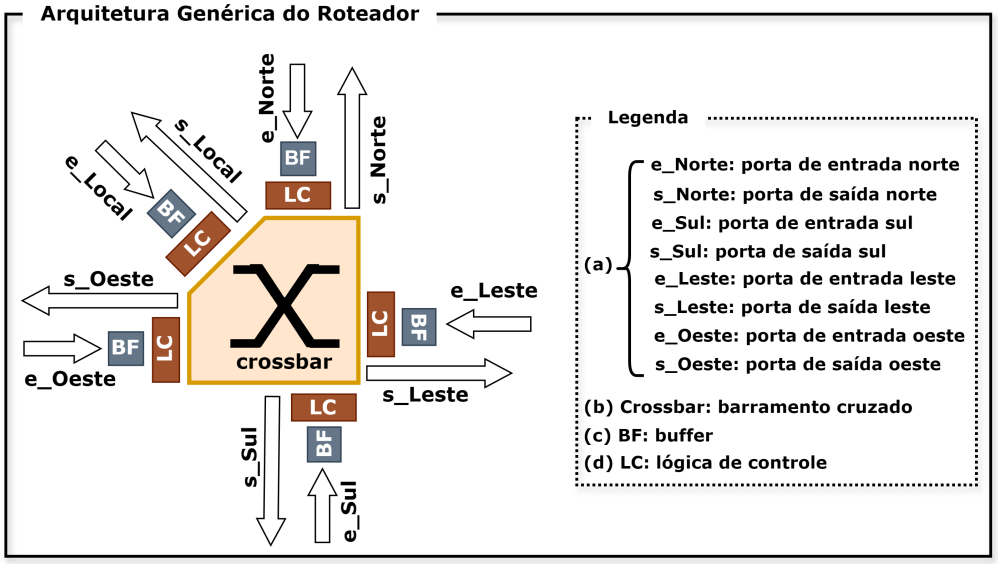

Figura 2.2 – Microarquitetura Genérica de um Roteador NoC.
(Adaptado de (TSAI et al., 2012))

Interligando estas portas, há uma matriz de interconexão do tipo *Crossbar* (Figura 2.2b), essencial para o desempenho eficiente da rede, suportando comunicações paralelas e flexíveis, reduzindo bloqueios e gerenciando eficientemente as solicitações de transferência de dados entre os PEs. Cada *link* possui um *buffer* (Figura 2.2c) para armazenamento temporário e uma unidade de lógica de controle (Figura 2.2d), que executa políticas de roteamento e arbitragem (SANTHOSH, 2015). Esses detalhes da arquitetura do roteador NoC destacam a importância dos

sistemas de interconexão, especialmente no controle de fluxo, que gerencia a alocação de recursos como *buffers* e *links*, essenciais para minimizar congestionamentos e maximizar a eficiência da transmissão de dados (OGRAS; MARCULESCU, 2006; AUSAVARUNGNIRUN; MUTLU, 2021).

2.2.2 Forma Geral do Pacote

Um pacote em uma NoC, ilustrado na Figura 2.3, é dividido em unidades menores chamadas *flits* (unidades de controle de fluxo). Cada *flit* desempenha um papel específico na transmissão de dados e é composto por três tipos principais:

Head Flit: Contém informações importantes, como endereço de origem, endereço de destino e dados de congestionamento. Essas informações são essenciais para o roteamento correto e eficiente dos pacotes através da rede.

Body Flit: Transporta os dados principais da mensagem. É o componente responsável por carregar a carga útil real da comunicação.

Tail Flit: Inclui informações de término da mensagem, como o fim dos dados e o número de sequência do *flit*, garantindo que os dados sejam reconstruídos na ordem correta no destino.

Essa estrutura modular dos *flits* permite que os dados sejam transmitidos de forma eficiente entre os nós da rede, com cada *flit* carregando partes específicas da informação necessária para a comunicação. A arquitetura do *flit* assegura uma transmissão de dados organizada, com controle preciso de fluxo e congestionamento, essencial para manter o desempenho e a integridade dos dados na rede NoC.

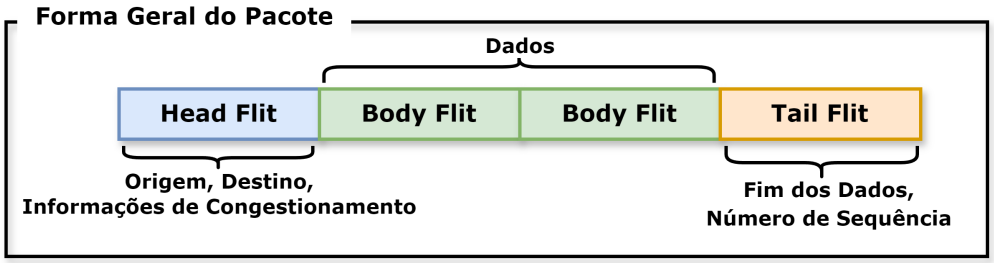

Figura 2.3 – Forma Geral do Pacote NoC.
(Adaptado de (ABBAS NOREEN JAMIL; ABBAS, 2021))

2.2.3 Topologias Diretas Tradicionais

A escolha da topologia em projetos de NoCs é essencial, afetando diretamente as medidas de desempenho descritas na subseção 2.5. Ela envolve a estrutura da rede e a organização de seus componentes, como roteadores e *links* de comunicação (ALIMI et al., 2021; KALITA et al., 2016). As topologias podem ser classificadas como diretas ou indiretas, regulares ou irregulares, homogêneas ou heterogêneas, além de serem bidimensionais (2D) ou tridimensionais (3D).

As topologias diretas, que são mais relevantes para o projeto apresentado nesta tese, conectam cada roteador a um elemento de processamento. Exemplos de topologias diretas incluem *Mesh*, *Torus*, *Ring*, *Octagon*, *Binary Tree* e *Star* (BABOLI; HUSIN; MARSONO, 2014; BABAIE et al., 2011; TATAS et al., 2014; KREUTZ et al., 2005; BJERREGAARD; MAHADEVAN, 2006; COTA; AMORY; LUBASZEWSKI, 2012), conforme ilustrado na Figura 2.4.

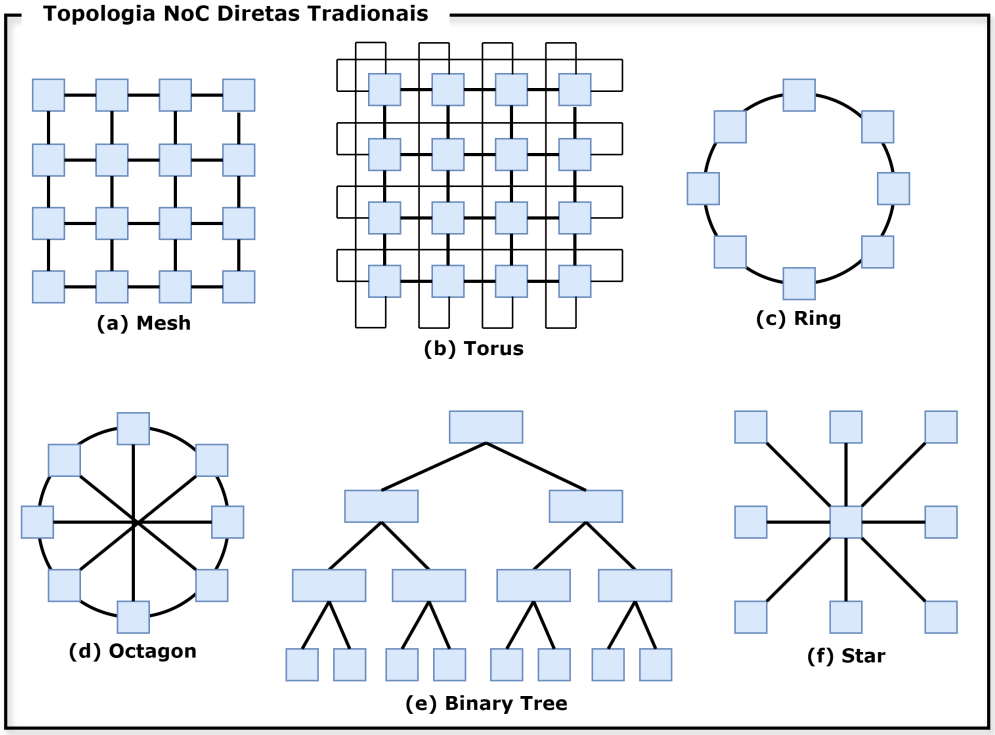

Figura 2.4 – Exemplos de Topologias NoC - Diretas Tradicionais: (a) *Mesh*, (b) *Torus*, (c) *Ring*, (d) *Octagon*, (e) *Binary Tree* e (f) *Star* (Adaptado de (PHING et al., 2017)).

Portanto, antes de detalhar algumas dessas principais topologias que se enquadram nas categorias mencionadas, é necessário entender alguns conceitos fundamentais. O *Router Degree* indica o número de conexões de um roteador, essencial para a análise da rede. As topologias são regulares quando todos os nós possuem o mesmo número de conexões, e irregulares quando há variações nesse número (KAMAL; GOYAL; NEHRA, 2012). Por exemplo, roteadores em topologias *torus* têm uniformemente quatro conexões, diferindo das *mesh*, em que os *Degrees* variam: dois nos cantos, três nas bordas e quatro no interior (AGHAEI et al., 2020).

Além disso, o *hop count* e o *diameter* da NoC, que representam o máximo de *hops* entre dois pontos, são utilizados para avaliar a eficiência e o potencial de atraso da rede. Topologias maiores tendem a aumentar a *link complexity*, impactando diretamente no custo e na complexidade do *hardware* (PARKHANI; TEHRE, 2016). Por fim, a *bisection width* mede a capacidade da rede de se dividir em duas partes iguais, sendo essencial para NoCs com alta demanda, pois indica a robustez e a capacidade de evitar bloqueios (TINETTI, 2008; ATAGOZIYEV, 2009).

A topologia *mesh*, exemplificada na Figura 2.4a, é preferida para conectar muitos núcleos em uma configuração regular, devido à sua eficiência em integrar uma vasta quantidade de núcleos (SADEGHI; NEZHAD; ZAVAREH, 2016). Em um arranjo 4x4 com 16 PEs, cada roteador liga-se a um recurso computacional e, geralmente, a quatro vizinhos, exceto nas bordas e cantos onde pode se ligar a três ou dois vizinhos, respectivamente, baseando-se em sua localização no *grid*. Destaca-se por proporcionar múltiplas rotas e excelente escalabilidade (KUMAR et al., 2002). Ademais, a tolerância a falhas é uma vantagem notável, permitindo a comunicação entre nós computacionais por diferentes caminhos. Porém, enfrenta o desafio do aumento no *diameter* conforme cresce o número de nós, o que pode aumentar a latência. Esse problema é agravado pela variação no *degree* dos roteadores e na largura de banda, particularmente nos roteadores de canto, borda e internos (RANE; GAD; PANEM, 2021).

Muito semelhante à estrutura da topologia *mesh*, mas oferecendo um *diameter* menor, a topologia *torus* tem como objetivo mitigar o problema de aumento de *diameter* observado em *grids*. Essa abordagem é alcançada por meio da extensão das conexões, como ilustrado na Figura 2.4b (CHANG; YUBAI; SONG, 2008).

Emprega-se o uso de *wrap-around channels* para a conexão de roteadores nas bordas opostas, o que contribui para a ampliação da *bisection width* e a diminuição no número médio de *hops*. Entretanto, essa configuração pode acarretar latências elevadas devido ao comprimento desses canais (CHEN; LI; GILLARD, 2011; BHANU et al., 2018).

No formato de um círculo, a topologia *ring*, ilustrada na Figura 2.4c, permite a conexão de cada nó computacional a dois outros roteadores através de *links*. Independentemente do número de nós, cada um possui duas portas. Devido ao menor número de canais, essa topologia tende a enfrentar congestionamentos, impactando sua eficiência (KAMATH; SAXENA; TALAWAR, 2015). Os *links* mantêm comprimentos uniformes, simplificando o *design* da rede. Contudo, a necessidade de canais virtuais duplica os requisitos de *buffers*. O roteamento pode ser realizado em sentido horário ou anti-horário, mas essa configuração limita a escalabilidade e a *bisection width* (KAMAL; GOYAL; NEHRA, 2012).

A topologia *octagon*, ilustrada na Figura 2.4d, é semelhante à *ring*, composta por oito nós computacionais e doze *links* bidirecionais. Cada nó é conectado aos nós precedente e subsequente, permitindo comunicações de dois *hops* entre pares de nós (KARIM; NGUYEN; DEY, 2002). Isso significa que cada nó possui conexão direta com três nós vizinhos e pode alcançar todos os outros nós em, no máximo, dois *hops*. Essa topologia utiliza um roteamento simples de caminho mais curto e oferece uma largura de banda agregada maior em comparação com a topologia de barramento compartilhado. Além disso, pode ser expandida para suportar *designs* maiores, resultando em melhor escalabilidade (BHOWMIK et al., 2020).

Outra topologia, a *Binary Tree*, apresentada na Figura 2.4e, organiza-se hierarquicamente com um nó computacional "raiz"no topo e vários outros nas "folhas"na base, facilitando a comunicação para fluxos de dados concentrados. O nó raiz é um ponto crítico, e sua falha impacta toda a rede, enquanto a complexidade aumenta com o tamanho da rede (KHAN; ANSARI, 2011; JEANG; JOU; HUANG, 2005). Roteadores intermediários atuam como pontos de encaminhamento, ligando os roteadores de folha aos clientes e promovendo uma ampla largura de banda. No entanto, essa topologia exige mais roteadores para conectar relativamente poucos nós, resultando em uma alta razão de roteadores para clientes e um *layout* interconectado complexo (VIPIN; KALOMIROS, 2019).

Por fim, a topologia *star*, ilustrada na Figura 2.4f, centraliza a gestão da rede em um nó central, com os nós periféricos apresentando um *degree* de conectividade de um, enquanto o nó central tem um *degree* correspondente ao total de nós conectados (SONG; MA; DALEI, 2008). O *diameter* constante em dois facilita a comunicação com um mínimo de dois *hops*, independentemente do tamanho da rede. Essa estrutura destaca-se pela comunicação simples e eficiente. No entanto, a falha do nó central pode incapacitar toda a rede, e o aumento no número de nós pode causar gargalo no nó central, afetando a eficiência da rede (YU et al., 2023).

2.3 Estratégias de Roteamento em NoCS

Para uma comunicação eficaz em NoCs, é essencial selecionar estratégias de roteamento adequadas. Além da topologia e das técnicas de controle de fluxo, que otimizam a entrega de pacotes, diversas estratégias são exploradas para otimizar o tráfego de dados, cada uma com suas particularidades, vantagens e limitações (UMA; SAROJADEVI; SANJU, 2019).

O *Deterministic Routing* se destaca por sua simplicidade, adotando sempre a mesma rota entre uma fonte e um destino, independente das condições de tráfego. Um exemplo clássico é o *Dimension Order Routing*, que prioriza a movimentação do pacote primeiramente na direção X e, em seguida, na direção Y (XY). Embora previsível, essa abordagem não considera as variações de tráfego, podendo resultar em congestionamentos (UMA; SAROJADEVI; SANJU, 2019; ALIMI et al., 2021). Além do roteamento XY, outras variantes determinísticas incluem *West-First* (WF), *Negative-First* (NF) e *Odd-Even* (OE), que oferecem diferentes abordagens para minimizar congestionamentos específicos (PHING et al., 2018; AHMAD; SETHI, 2020).

O *Adaptive Routing* se ajusta às condições atuais da rede, escolhendo caminhos com base no estado de congestionamento. Essa categoria se divide em *Minimal Adaptive Routing*, que mantém o pacote o mais próximo possível do destino, e *Non-minimal Adaptive Routing*, que permite desvios para evitar áreas congestionadas (RANTALA et al., 2006). Dentre as estratégias adaptativas, destacam-se *North-Last* (NL) e *XYX*, que oferecem maior flexibilidade e potencial redução de congestionamento. Essas estratégias permitem uma utilização mais eficiente da

rede e um equilíbrio de carga, embora aumentem a complexidade na tomada de decisões e na manutenção da ordem dos pacotes (UMA; SAROJADEVI; SANJU, 2019; PHING et al., 2018; AHMAD; SETHI, 2020).

2.3.1 Modelo de Movimentos

O modelo de movimentos *Turn Model* é uma estratégia muito utilizada no desenvolvimento de algoritmos de roteamento para NoCs, buscando evitar os desafios já citados anteriormente (CHEMLI; ZITOUNI, 2015). Originados por dependências circulares de recursos, *deadlocks* são mitigados pelo modelo através da restrição de certas curvas no deslocamento dos pacotes, desfazendo ciclos de dependência. Este modelo visa a um equilíbrio, limitando proibições ao necessário, mantendo caminhos disponíveis entre todos os pares de nós, e otimizando a adaptabilidade dos algoritmos de roteamento (BROWN, 2013).

Essas estratégias de roteamento formam o *framework* para o movimento de dados, com algoritmos específicos implementados para otimizar a eficiência, oferecendo soluções adaptáveis para a integridade do tráfego de dados.

2.3.2 Algoritmos de Roteamento para *Mesh* 2D

Segundo Brown (2013), dentro de uma topologia de rede em malha 2D, um pacote de dados pode se mover em quatro direções básicas: para cima (Norte), para baixo (Sul), para a esquerda (Oeste) e para a direita (Leste). Um movimento, que também pode ser considerado uma curva, ocorre quando um pacote altera sua direção de deslocamento. Por exemplo, o pacote pode mudar sua direção de leste para norte, formando um ângulo reto ($\rightarrow \uparrow$), ou de sul para oeste ($\leftarrow \downarrow$) (BROWN, 2013).

A legenda nas Figuras 2.5, 2.6 e 2.7 representa os elementos importantes da movimentação do pacote. O Roteador de Origem (em verde) indica o ponto inicial de envio do pacote. O Roteador Intermediário (em azul claro tracejado) representa a posição atual do pacote durante a movimentação. O Roteador de Destino (em laranja) indica o ponto final de entrega do pacote. O Roteador Ocupado (em vermelho) denota um nó que está ocupado e não pode ser utilizado para o rote-

amento do pacote. O Link Percorrido (seta roxa) representa a trajetória seguida pelo pacote ao se deslocar pela rede.

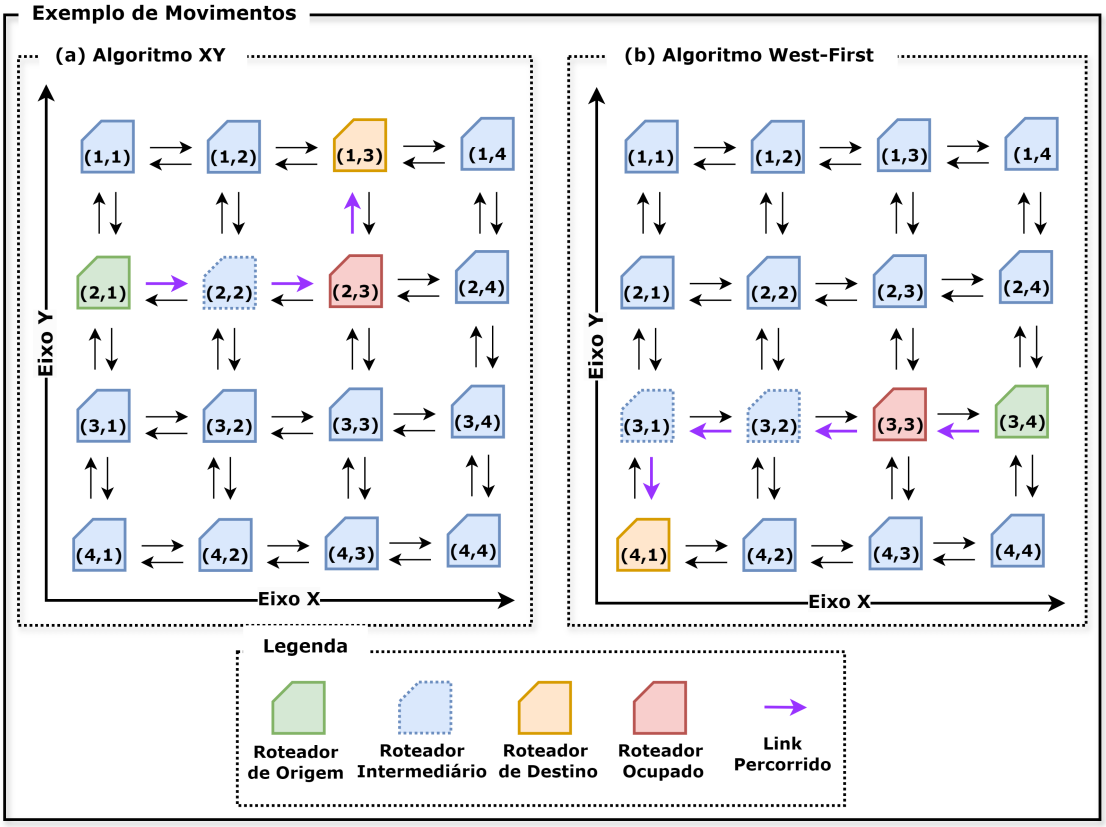

Figura 2.5 – Exemplo de Movimentos para os Algoritmos XY e *West-First*.

Na Figura 2.5, são ilustrados os modelos de movimentos dos algoritmos determinísticos. O *XY*, ilustrado na Figura 2.5a, prioriza a movimentação do pacote primeiramente na direção X e, em seguida, na direção Y. Neste exemplo, o roteador de origem está na posição (2,1) e o destino na posição (1,3). O pacote move-se horizontalmente da posição (2,1) para (2,3) e então verticalmente para a posição (1,3). Como não há um roteador ocupado, ele pode continuar (MAHENDRA; GAIKWAD; PATRIKAR, 2016; DHARWAD, 2013).

O outro exemplo, representado na Figura 2.5b, é o algoritmo *West-First*, uma variante do roteamento determinístico que evita movimentos para o leste até que todos os movimentos para o oeste tenham sido concluídos. No exemplo, o pacote parte do roteador na posição (3,4) com destino ao roteador na posição (4,1). A

trajetória inicial do pacote segue para o oeste até a posição (3,3), mas encontra um roteador ocupado. Nesse ponto, o algoritmo aguarda a liberação do roteador antes de continuar para o oeste, depois se move para o sul até o destino na posição (4,1) (MAHENDRA; GAIKWAD; PATRIKAR, 2016; DHARWAD, 2013).

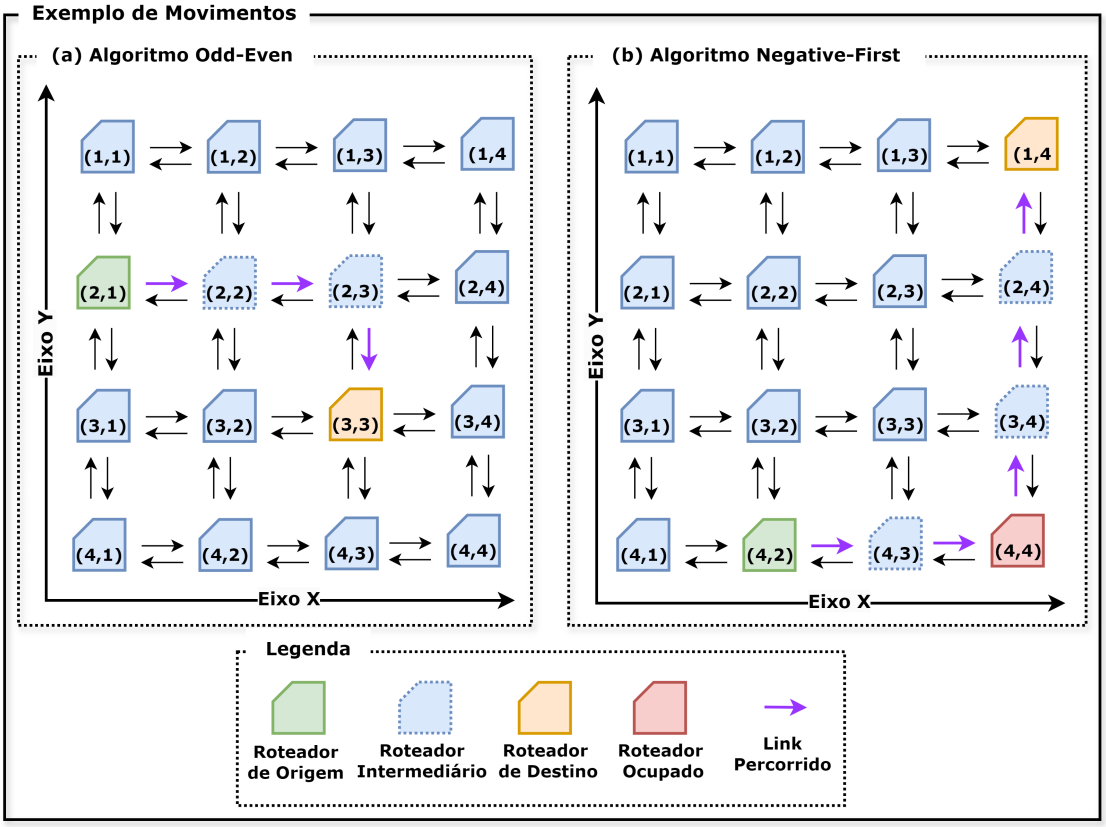

Figura 2.6 – Exemplo de Movimentos para os Algoritmos *Odd-Even* e *Negative-First*.

Na Figura 2.6, são ilustrados os modelos de movimentos dos algoritmos *Odd-Even* e *Negative-First*. O *Odd-Even*, ilustrado na Figura 2.6a, permite decisões de roteamento baseadas na paridade das coordenadas dos nós, oferecendo um balanceamento de carga melhorado. Neste exemplo, o roteador de origem está na posição (2,1) e o destino na posição (3,3). O pacote move-se horizontalmente da posição (2,1) para (2,3) e depois verticalmente para a posição (3,3), seguindo as regras de paridade. Como não há um roteador ocupado, ele pode continuar sem interrupções (ZHANG et al., 2009a; ZHU; PANDE; GRECU, 2007).

O outro exemplo, representado na Figura 2.6b, é o algoritmo *Negative-First*, que prioriza movimentos em direções negativas (oeste e sul) antes de permitir movimentos em direções positivas (leste e norte). No exemplo, o pacote parte do roteador na posição (4,2) com destino ao roteador na posição (1,4). A trajetória inicial do pacote segue para o leste até a posição (4,4), mas encontra um roteador ocupado. Nesse ponto, o algoritmo aguarda a liberação do roteador antes de continuar para o norte e depois para o destino na posição (1,4) (ZHANG et al., 2009a; ZHU; PANDE; GRECU, 2007).

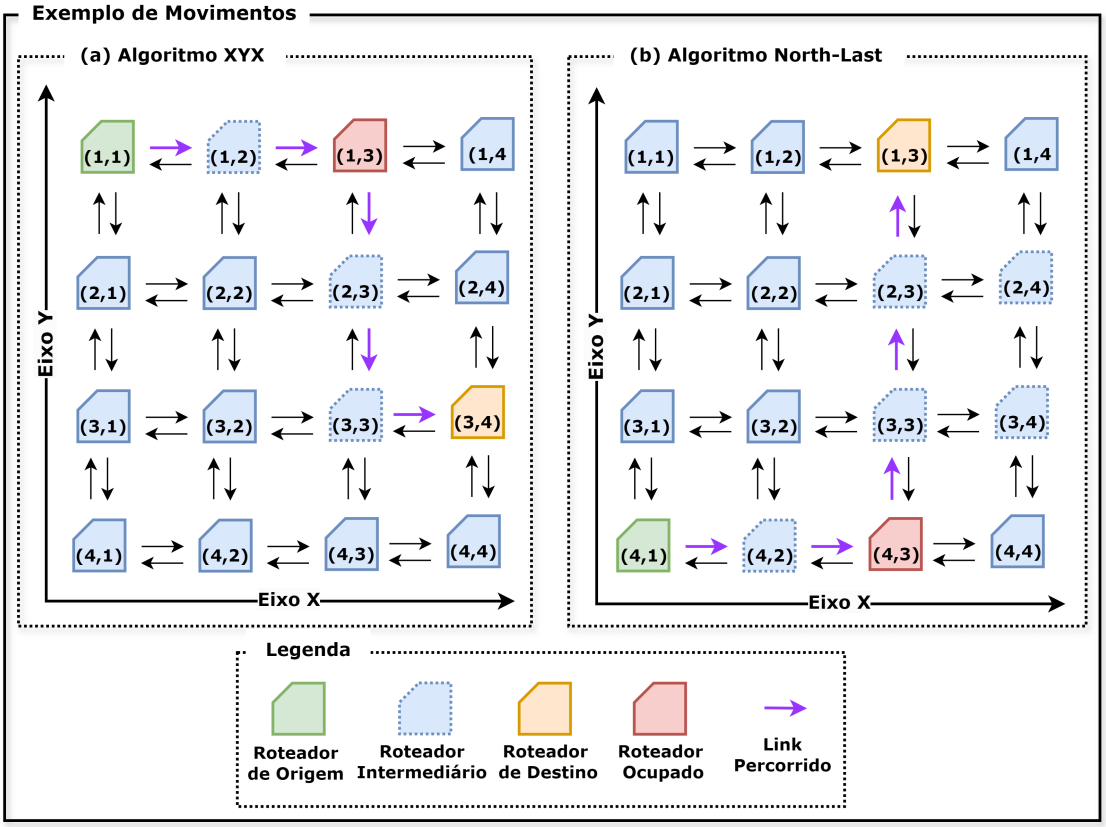

Figura 2.7 – Exemplo de Movimentos para os Algoritmos XYX e *North-Last*.

Na Figura 2.7, são ilustrados os modelos de movimentos dos algoritmos adaptativos XYX e *North-Last*. O XYX, ilustrado na Figura 2.7a, é uma variação do roteamento XY que permite movimentos em ambas as direções X antes de mover na direção Y. Neste exemplo, o roteador de origem está na posição (1,1) e o destino na posição (3,4). O pacote move-se horizontalmente da posição (1,1) para (1,3),

mas encontra um roteador ocupado na posição (1,3). Nesse ponto, o algoritmo muda de direção conforme necessário, movendo-se para (2,3) e então para (3,3), e finalmente para (3,4).

O outro exemplo, representado na Figura 2.7b, é o algoritmo *North-Last*, que proíbe movimentos para o norte até que todos os movimentos em outras direções tenham sido concluídos. No exemplo, o pacote parte do roteador na posição (4,1) com destino ao roteador na posição (1,3). A trajetória inicial do pacote segue para o leste até a posição (4,3), mas encontra um roteador ocupado. Nesse ponto, o algoritmo muda de direção conforme necessário, movendo-se para o norte até a posição (3,3) e depois continua para o norte até o destino na posição (1,3) (MAHENDRA; GAIKWAD; PATRIKAR, 2016; DHARWAD, 2013).

2.4 Mapeamento de Aplicações em NoCs

O mapeamento de aplicações em arquiteturas *many-core* com NoC, ilustrado na Figura 2.8, representa um desafio que demanda a alocação eficiente de elementos de processamento e o gerenciamento adequado da comunicação (BENCHEHIDA et al., 2021). As estratégias de *Static Mapping* (XU; TANIGUCHI; TOMIYAMA, 2017), *Dynamic Mapping* (DALZOTTO et al., 2021) e *Hybrid Mapping* (POURMOHSENI et al., 2020) são fundamentais para otimizar o desempenho, reduzir o consumo de energia e aumentar a confiabilidade. No *Static Mapping*, os parâmetros definidos inicialmente são adequados a cargas de trabalho constantes, enquanto a *Heuristic Search* facilita a obtenção de soluções eficientes com menor demanda de recursos computacionais (AMIN et al., 2020).

Por outro lado, a busca exata (*Exact Search*) se destaca pela aplicação de técnicas rigorosas para encontrar soluções ótimas, frequentemente implementada através de (*Mathematical Programming-Based*). Este enfoque visa a otimização completa, enfrentando, porém, o desafio de sua alta complexidade computacional. Em contraste, o mapeamento dinâmico oferece flexibilidade, adaptando-se a mudanças nas demandas em tempo real, sendo adequado para ambientes com cargas de trabalho variáveis (SAMBANGI et al., 2023).

No contexto do mapeamento híbrido, a subdivisão baseada em busca (*Search-based*) emprega algoritmos de busca para otimizar a alocação de tarefas (SINGH et

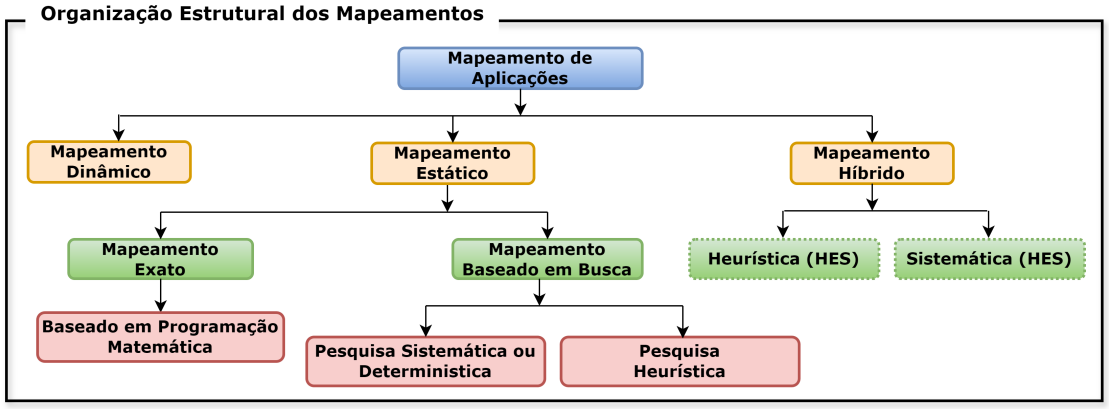

Figura 2.8 – Organização Estrutural Resumida dos Mapeamentos de Aplicações em Arquiteturas *Many-core*. (Adaptado de (AMIN et al., 2020))

al., 2017). A abordagem híbrida exata e de busca (*Hybrid Exact & Search (HES)*) combina heurísticas (*HES Heuristic*) e métodos sistemáticos (*HES Systematic*), buscando um equilíbrio entre precisão e eficiência operacional (AMIN et al., 2020).

Após discutir o mapeamento de aplicações em arquiteturas *many-core*, a atenção se volta para técnicas de exploração do espaço de *design* desses sistemas. Essa etapa, diretamente ligada aos conceitos anteriores, propõe um método sistemático para otimizar a configuração dessas arquiteturas.

2.5 Métricas de Desempenho para NoCs

No campo de interconexão e comunicação em NoCs, a análise detalhada de requisitos fundamentais é essencial para compreender os benefícios e desafios associados a essas redes. Estudos especializados frequentemente avaliam as métricas de desempenho e os custos relacionados à rede, utilizando métodos específicos para explorar o impacto de diversos parâmetros de projeto. Entre as métricas mais utilizadas na literatura estão a *latency*, a *throughput*, a *bandwidth*, a *fault tolerance* e o *energy consumption*, todas fundamentais para determinar a eficiência e a eficácia da rede (MANDAL et al., 2020). Essas métricas são apresentadas em maior detalhe nas subseções seguintes.

2.5.1 Latência

Um aspecto crítico no estudo de NoCs é a avaliação de seu desempenho, frequentemente medido pela latência de pacotes. Segundo (MATOUSSI, 2021), a latência de pacotes é definida como o tempo necessário para um pacote viajar de sua fonte até seu destino na rede. Esta métrica é fundamental, pois impacta diretamente o desempenho geral do sistema em que a NoC está implantada. Portanto, a latência de um pacote individual na rede é calculada por uma fórmula específica que leva em conta vários fatores, conforme descrito na Equação 1 abaixo:

$$Lat_{pkti} = (P_{lat_{pkti}} \times \text{nbrhops}) + ((L-1) \times FT) + Wait_{pkti} \quad (1)$$

Onde:

a) $P_{lat_{pkti}}$ representa o atraso físico, influenciado por aspectos físicos da rede, como atrasos nos roteadores e nos *links*;

b) nbrhops é o número de roteadores que o pacote atravessa, um valor determinado pelo algoritmo de roteamento utilizado;

c) FT é o tempo de transmissão de um *flit*, a unidade básica de transferência em uma NoC;

d) L indica o tamanho do pacote em termos de *flits*;

e) $Wait_{pkti}$ é o atraso de contenção, que ocorre devido a fatores como a competição por canais compartilhados e a sobrecarga dos *buffers* dos roteadores.

Esta formulação da latência de pacotes, como definida na Equação 1, captura tanto o atraso de *hop-count*, que inclui atrasos intra e inter-roteadores, quanto o número de *hops*, e o atraso de contenção, relacionado à competição por recursos e ao congestionamento de *buffers*. O termo $(L-1) \times FT$ considera que o primeiro *flit* do pacote é enviado imediatamente, enquanto os *flits* subsequentes $(L-1)$ adicionam tempo de transmissão adicional.

Esta abordagem assegura que a latência inclua todos os aspectos significativos do atraso na rede, fornecendo uma medida abrangente do desempenho. É importante destacar que, nesse estudo específico, os autores focam na latência definida por essa equação, considerando tanto os atrasos físicos quanto os de contenção,

mas abstraem o tempo de espera nas filas de injeção (ou tempo de enfileiramento na fonte) (DUMITRASCU et al., 2006).

2.5.2 Taxa de Transferência

A taxa de transferência representa um aspecto fundamental que define a eficiência do sistema, ou seja, a carga máxima que pode ser utilizada para determinar a largura de banda agregada do sistema. Em termos simples, ela pode ser vista como a soma da taxa de transferência de cada núcleo ou nó na rede, que representa a quantidade de bits que chega ao núcleo durante um intervalo de tempo específico (SUBOH et al., 2010). A quantidade de núcleos selecionados como destinos é representada pelo acumulado de *flits* que chegam a esse núcleo até um dado momento. Quando a carga de novos *flits* se aproxima do limite de saturação da taxa de transferência, a latência aumenta exponencialmente. Essa relação entre a carga de tráfego e a latência é um aspecto crítico no design de sistemas NoCs (LIU; ZHENG; TENHUNEN, 2003).

Uma maneira comum de quantificar a taxa de transferência é através da Equação 2 a seguir:

$$Th = \frac{Rf}{Nn \times Nc} \tag{2}$$

onde:

a) Rf representa o número total de *flits* recebidos;

b) Nn é o número total de nós computacionais na rede;

c) Nc é o número de ciclos de *clock* decorridos desde o primeiro *flit* gerado até o último *flit* recebido.

Assim, essa equação calcula a taxa de transferência média ao longo de um período específico, fornecendo uma medida precisa da capacidade de comunicação do sistema. Destaca a eficiência da rede em processar e transmitir dados, sendo essencial para avaliar o desempenho global de sistemas NoCs.

2.5.3 Largura de Banda

A *largura de banda* é um conceito fundamental que define a taxa máxima de transmissão de dados, geralmente medida em *bits* por segundo (*BPS*). Ela determina a velocidade com que os dados são transmitidos através de uma rede. Na sua avaliação, considera-se a soma dos *bits* de dados, incluindo o cabeçalho, a carga útil e o terminador da mensagem, abrangendo todos os aspectos da transmissão. Conforme explicado por Hennesey e Patterson (2012), uma maior *largura de banda* implica uma velocidade de conexão mais rápida, permitindo a transmissão de uma maior quantidade de dados simultaneamente, o que aumenta a eficiência da rede (HENNESSY; PATTERSON, 2012).

2.5.4 Tolerância a Falhas

A resiliência de um sistema diante de falhas é um aspecto crítico da confiabilidade. Um sistema é considerado tolerante a falhas quando consegue manter sua funcionalidade mesmo na presença de falhas. Para alcançar tal resiliência, são empregadas diversas técnicas de tolerância a falhas, cuja complexidade varia conforme os requisitos de confiabilidade. Essas técnicas incluem redundância espacial, utilizando componentes físicos distintos para duplicar sinais de dados, e redundância temporal, em que os mesmos componentes físicos são usados para gerar sinais de dados idênticos em momentos diferentes (BONNEY et al., 2016).

Em arquiteturas *many-core*, a (*Graceful Degradation*) é uma estratégia de tolerância a falhas. Essa técnica se beneficia da redundância inerente a essas arquiteturas, que possuem múltiplos núcleos idênticos, bancos de memória e roteadores NoC, permitindo que o sistema continue operando, embora com capacidade reduzida, mesmo diante de falhas permanentes causadas pela variabilidade do processo de fabricação e pelo envelhecimento acelerado dos componentes (BONNEY et al., 2016).

2.5.5 Estimativa para o Consumo de Energia

Os modelos de energia são fundamentais no projeto e na otimização de sistemas baseados em *NoC*. O simulador ORION destaca-se como um modelo arquitetural

para a estimativa de potência e área em sistemas *NoC*. Com três versões (WANG et al., 2002; KAHNG et al., 2009; KAHNG; LIN; NATH, 2015), cada uma introduz melhorias e novas funcionalidades. As versões 2.0 e 3.0, por exemplo, estão alinhadas com *designs* reais de redes de comunicação. Sua precisão é validada ao ser comparada com o consumo real de energia em projetos complexos, como os da Intel com 80 núcleos (MATTSON; WIJNGAART; FRUMKIN, 2008). Portanto, esse simulador é essencial para estimar o consumo de energia e a área ocupada por roteadores, influenciando diretamente a eficiência energética e o desempenho espacial dos sistemas.

Outra abordagem é o *(Complementary Metal-Oxide-Semiconductor)* (CMOS), essencial para simulações ao nível de transistor e suas implicações nas estimativas de energia e desempenho. Esses modelos fornecem informações detalhadas sobre as características elétricas de componentes semicondutores, sendo indispensáveis para simulações precisas em NoC . Eles ajudam a compreender como o consumo de energia, o tempo de propagação e a capacidade de carga dos transistores afetam o desempenho geral do sistema (DOMAN, 2012).

Outro método importante é o de energia por *bit (Ebit)*. Os pesquisadores do trabalho (HU; MARCULESCU, 2003), inspirados no modelo proposto pelos autores do artigo (YE; BENINI; MICHELI, 2002) para roteadores de NoCs, desenvolveram um modelo de consumo de energia específico para NoCs. No modelo original, a energia por *bit (Ebit)* é definida como a energia consumida ao transportar um *bit* de dados através do roteador, incluindo a energia do próprio roteador (E_{Sbit}), do *buffer* (E_{Bbit}) e dos fios de interconexão (E_{Wbit}). Contudo, para NoCs, considera-se que a energia consumida pelos *buffers* e pelos fios internos é praticamente irrelevante em comparação à energia utilizada nos *links* entre os nós computacionais. As Equações 3, 4, 5 e 6 apresentam uma forma de calcular esse consumo de energia, ajustado para NoCs e topologias *mesh* 2D.

$$E_{bit} = E_{Sbit} + E_{Bbit} + E_{Wbit} + E_{Lbit} \tag{3}$$

onde:

a) E_{Sbit} representa a energia consumida pelo roteador;

b) E_{Bbit} é a energia consumida pelo *buffer* (considerada negligenciável em

NoCs);

c) E_{Wbit} indica a energia dos fios de interconexão (considerada negligenciável em NoCs);

d) E_{Lbit} é a energia consumida nos *links* entre nós computacionais.

$$E_{bit} = E_{Sbit} + E_{Lbit} \tag{4}$$

$$E_{bit}^{ti,tj} = n_{hops} \times E_{Sbit} + (n_{hops} - 1) \times E_{Lbit} \tag{5}$$

$$E_{total} = \sum_{i=1}^{N^2} [E_{bit}^{ti,tj}] \tag{6}$$

É importante destacar que o modelo se concentra na energia consumida nos *links* de comunicação, sem considerar a energia média consumida pelos recursos computacionais dos nós da rede. As abordagens apresentadas enfatizam a importância da comunicação nas NoCs, um ponto-chave no consumo de energia. Ao mesmo tempo, o gerenciamento de energia dos componentes computacionais é realizado de forma distinta. Após a exposição da estrutura geral das NoCs, o foco se direciona ao mapeamento de aplicações, um elemento crítico na otimização de recursos e na minimização da latência de comunicação, essencial para atender às demandas de desempenho e eficiência energética em arquiteturas *many-core* (SINGH et al., 2013; SINGH et al., 2017).

2.6 Exploração do Espaço de *Design* (DSE)

Na busca por soluções otimizadas para sistemas *many-core* baseados em NoCs, a *Design Space Exploration (DSE)* é fundamental. Esse processo envolve encontrar soluções de projeto que melhor atendam aos requisitos desejados a partir de um espaço de pontos de *design* provisórios (CARDOSO; COUTINHO; DINIZ, 2017b). A DSE é complexa devido à necessidade de avaliar conjuntamente opções de projeto de hardware e software. Neste trabalho, a busca por soluções considera parâmetros como complexidade do *DAG*, topologia da NoC e algoritmos de mapeamento de tarefas e roteamento de pacotes. A diversidade e complexidade desse processo

demandam métodos e ferramentas de automação que simplifiquem as etapas de geração, avaliação e seleção de opções de projeto (KANSAKAR; MUNIR, 2018).

Para encontrar soluções ótimas, tanto para objetivos únicos quanto para otimização multiobjetivo, a DSE utiliza diversos algoritmos. Para otimização de objetivo único, são empregados algoritmos como *Simulated Annealing* (SA) (BERTSIMAS; TSITSIKLIS, 1993), *Tabu Search* (TS), *Genetic Algorithms* (GA) (HOLLAND, 1992), *Ant Colony Optimization* (ACO) (DORIGO; CARO, 1999), *Particle Swarm Optimization* (PSO) (EBERHART; KENNEDY, 1995), *Integer Linear Programming* (ILP) (PAN; PAN, 2014), *Random Search* (KARP, 1991), *Branch and Bound* (BB) (BOUKEDJAR; LALAMI; EL-BAZ, 2012) e *Near-optimal Mapping* (NMAP) (TRAN; BAAS, 2012) (CARDOSO; COUTINHO; DINIZ, 2017a; KANSAKAR; MUNIR, 2018).

Os métodos de DSE integram técnicas de otimização e automação, visando atender aos requisitos de desempenho e eficiência energética, entre outros. Eles são essenciais diante da crescente complexidade das arquiteturas *many-core* e embarcadas, permitindo um equilíbrio eficaz entre parâmetros de interesse por meio de uma exploração guiada do espaço de projeto (KANSAKAR; MUNIR, 2018).

2.7 Considerações finais

Neste capítulo, foram abordadas as bases teóricas essenciais para entender arquiteturas *many-core*, com foco em NoCs, sistemas de interconexão e estratégias de mapeamento e roteamento. Discutiram-se algoritmos de roteamento e medidas de desempenho, como latência, taxa de transferência, largura de banda, tolerância a falhas e métodos para estimar o consumo de energia. Esses conceitos fornecem a base teórica necessária para compreender a otimização dessas arquiteturas. Foi apresentado o conceito de exploração de espaço de *design*, integrando os tópicos abordados. No próximo capítulo, serão discutidos trabalhos relacionados aos objetivos deste estudo, destacando suas intersecções conceituais e práticas, situando esta pesquisa no contexto atual da área.

Capítulo 3

Trabalhos Relacionados

Neste capítulo, será apresentada uma revisão dos principais trabalhos científicos encontrados na literatura relacionados aos objetivos do projeto de pesquisa apresentado nesta tese, em particular aqueles que tratam de simuladores específicos para NoCs

.

Sistemas *many-core* baseados em arquiteturas NoC têm contribuído significativamente para o avanço da computação de alto desempenho. Esses sistemas são de grande interesse para a proposição de ferramentas de simulação que apoiem as atividades de projeto e implementação de plataformas desse tipo. Diversos simuladores voltados para sistemas *multi-core* ou redes em geral são encontrados na literatura, como OpenPiton (BALKIND et al., 2016), Gem5 (BINKERT et al., 2011), Graphite (MILLER et al., 2010), NS2 (ISSARIYAKUL; HOSSAIN, 2011), NS3 (RILEY; HENDERSON, 2010), entre outros. Embora possam ser utilizados em estudos similares aos desta tese, são ferramentas de uso geral e não aderem facilmente aos objetivos específicos deste trabalho.

O trabalho desenvolvido por Pedrino e Tempesti, fruto da colaboração entre grupos de pesquisa da UFSCar e da Universidade de York trata da investigação do uso de sistemas inteligentes para mapeamento eficiente de aplicações em arquiteturas *many-core* (PEDRINO; LIMA; TEMPESTI, 2019). Tal abordagem explora a

Mathematical Morphology (MM), uma técnica para a análise de estruturas geométricas. Tendo como base a decomposição de operadores complexos em sequências de operadores mais simples para reduzir requisitos computacionais, destacando o *trade-off* entre qualidade do filtro morfológico e desempenho computacional. Utiliza um algoritmo evolutivo multiobjetivo para investigar esses *trade-offs*, gerando grafos de tarefas e mapeamentos que representam diferentes compromissos entre os objetivos, e fornecendo uma frente de Pareto de soluções de mapeamento para selecionar implementações que atendam a requisitos específicos da aplicação. Este projeto foi de grande valia para direcionar o desenvolvimento de alguns dos aspectos do trabalho apresentado nesta tese.

Por outro lado, algumas ferramentas descritas na literatura são especificamente voltadas para NoCs, sendo candidatas naturais para servirem como referência à proposta alternativa descrita nesta tese. O grupo de ferramentas descrito nas próximas seções oferece recursos razoavelmente homogêneos entre si, permitindo avaliações comparativas, especialmente no que diz respeito a aspectos como estratégias de mapeamento, algoritmos de roteamento, eficiência, confiabilidade e otimização de desempenho (KHAN et al., 2017b). O levantamento que se segue está subdividido em ferramentas, e para cada uma delas, diversos trabalhos de pesquisa são relatados, buscando evidenciar suas funcionalidades e usos típicos.

3.1 Simulador NoCTweak

O simulador NoCTweak (TRAN; BAAS, 2012) é uma ferramenta de código aberto, baseada em SystemC e C++, utilizada para explorar o desempenho e a eficiência energética em NoCs. É parametrizável, permitindo aos usuários configurar e simular diversos parâmetros de rede, incluindo o tipo de roteador, o tamanho da rede, o tamanho do buffer, o algoritmo de roteamento, a política de arbitragem, os estágios de pipeline, a tensão de alimentação, a frequência do clock, o padrão de tráfego, o comprimento do pacote, a taxa de injeção e os tempos de simulação e aquecimento. Esta ferramenta também engloba algoritmos de mapeamento de tarefas, como RANDOM e NMAP, que facilitam a criação de mapas de alocação. Os resultados estatísticos fornecidos pelo simulador incluem a latência média da rede, o *throughput*, a potência do roteador e a energia consumida por pacote transferido

(KHAN et al., 2017b).

Gawish et al. (2014) investigaram como a adaptação de algoritmos de roteamento convencionais pode minimizar falhas devido a variações no processo de fabricação. Utilizando o simulador NoCTweak, foi avaliada a eficácia desses algoritmos modificados. Experimentos com algoritmos como XY, *West-First*, *Negative-First* e *Odd-Even* mostraram uma redução significativa na taxa de falhas das NoCs, com o algoritmo West-First alcançando até 56% de redução na taxa de falhas, demonstrando a eficácia das modificações propostas (GAWISH; EL-KHARASHI; ABU-ELYAZEED, 2014).

Hassan et al. (2017) apresentaram uma extensão para o simulador NoCTweak, focada na simulação de agrupamento em NoC. Essa extensão permitiu agrupar múltiplos *clusters* em uma grade e configurar tipos de interconexão entre eles. A abordagem suportou diferentes algoritmos de roteamento para tráfego interno e externo aos *clusters*, incluindo fatores de atraso em *links inter-cluster* e um gerenciador de tráfego. Os resultados mostraram que o agrupamento pode reduzir o consumo médio de energia do roteador em cerca de 50%, evidenciando a eficácia dessa abordagem na avaliação de desempenho nesse contexto (HASSAN; MORGAN; EL-KHARASHI, 2017).

Gulzari et al. (2019) investigaram a topologia H-SMBFT para comunicação em chip. Este trabalho focou na arquitetura e nas características da H-SMBFT, comparando-a com as topologias BFT e SMBFT. Os resultados, segundo os autores, mostraram que a H-SMBFT superou suas antecessoras em termos de desempenho, área e dissipação de potência. Utilizando simulações, constatou-se uma redução significativa na complexidade do roteador, no número de *links* necessários e no caminho de roteamento, levando a melhorias no atraso médio, na área e no consumo de energia. A H-SMBFT foi analisada utilizando os simuladores ORION 3.0 e NoCTweak, considerando tanto tráfegos sintéticos quanto cargas de trabalho de aplicações do mundo real (GULZARI et al., 2019).

Sikandar et al. (2021) destacaram o *Sailfish Optimization Algorithm* (SFOA), um algoritmo meta-heurístico baseado na natureza para otimização do mapeamento de tarefas em NoCs. O foco deste trabalho foi minimizar a dissipação de energia nessas redes. Utilizando uma metodologia de agrupamento *k-nearest neighbor*, a abordagem demonstrou ser superior a outras técnicas similares. Para

avaliar a eficácia do algoritmo, o estudo empregou o simulador NoCTweak, analisando a latência média da rede do algoritmo em comparação com outros algoritmos heurísticos inspirados na natureza. Os resultados mostraram-se promissores, especialmente em grafos de tarefas de aplicativos de grande escala (SIKANDAR et al., 2021b).

Mehmood et al. (2022) aplicaram o algoritmo *Andean Condor Algorithm* (ACA) para mapeamento eficiente em NoCs. Ele focou no mapeamento de tarefas de aplicações em tempo real em múltiplos núcleos da rede, visando à otimização de desempenho e redução do consumo de energia. Inspirado no comportamento natural do condor andino, o algoritmo utilizou uma técnica baseada em *clustering* para uma convergência rápida. Os resultados experimentais mostraram que o ACA superou algoritmos do estado da arte em termos de custo de comunicação, latência média, taxa de transferência e consumo de energia, alcançando melhorias significativas em comparação com técnicas anteriores (MEHMOOD et al., 2022).

Ahmed e Baig (2023) apresentaram um algoritmo de roteamento ciente de congestionamento para NoCs, visando à eficiência na utilização de recursos e à redução da carga de trabalho em recursos individuais, diminuindo, assim, o consumo de energia. O simulador NoCTweak foi utilizado para a avaliação de desempenho, oferecendo resultados sobre latência média, taxa de transferência e consumo de energia do roteador em uma biblioteca CMOS de 65nm. Resultados experimentais indicaram que, embora o algoritmo proposto não tenha superado significativamente o algoritmo de *subnet*, ele se mostrou competitivo e eficaz. Sob padrões de tráfego transposto, o algoritmo demonstrou ser superior ao algoritmo de roteamento *Dy-Adaptive*, e em altas taxas de injeção de tráfego (0.75 *Flits/Node/Cycle*), ele se destacou novamente, mostrando robustez e eficácia (AHMED; BAIG, 2023).

3.2 Simulador Noxim

A ferramenta *Noxim* é um simulador NoC desenvolvido na Universidade de Catania, Itália (CATANIA et al., 2015). Embora não inclua algoritmos de mapeamento entre suas funcionalidades, de acordo com a literatura, é um dos mais utilizados até o momento presente. Baseado em *SystemC*, ele é projetado para avaliar NoCs em termos de taxa de transferência, latência e consumo de energia.

Oferecendo flexibilidade na simulação, o *Noxim* permite a customização de diversos parâmetros como tamanho da rede, tamanho do *buffer*, algoritmo de roteamento e estratégias de seleção. O simulador é compatível com topologias *mesh* 2D e suporta padrões de tráfego sintético. Entre os algoritmos de roteamento implementados estão XY, NF, WF, NL, OE e Dyad, totalmente adaptativo e baseado em tabela de pesquisa. Os resultados estatísticos incluem latência média da rede, taxa de transferência e consumo de energia (KHAN et al., 2017b).

Afsharpour et al. (2016) apresentaram um algoritmo eficiente de migração de tarefas projetado para *chips multi-core* e *many-core* baseados em *mesh*. O objetivo era reduzir os pontos de superaquecimento no *chip* e promover uma distribuição equilibrada de carga. Utilizando o simulador Noxim, o estudo evidenciou que o algoritmo proposto superou seus predecessores em diversos aspectos, alcançando melhorias notáveis de até 36% no desempenho, uma redução de 28% no consumo de energia, além de um controle de temperatura mais eficiente (AFSHARPOUR; PATOOGHY; FAZELI, 2016).

Kumar e Rao (2021) introduziram um algoritmo adaptativo de mapeamento de núcleos, projetado para otimizar o desempenho de sistemas heterogêneos em *chip*. O algoritmo demonstrou uma redução de até 18,2% no atraso de comunicação, um aumento de até 19,4% no *throughput* e uma diminuição de até 18,6% no consumo de energia (KUMAR; RAO, 2021).

Para concluir, Muhsen et al. (2023) apresentaram um modelo preditivo que combina *Artificial Neural Networks* (ANN) com otimização heurística. Utilizando o simulador Noxim, o estudo propôs uma metodologia eficiente para prever algoritmos de roteamento, facilitando uma configuração mais ágil dos sistemas MPSoC. Segundo os autores, o modelo híbrido GCAOA-ANN superou modelos anteriores, destacando-se na previsão de algoritmos de roteamento para ambientes NoC e ressaltando a importância das abordagens preditivas na otimização de sistemas em *chip* (MUHSEN et al., 2023).

3.3 Simulador BookSim

O BookSim é um simulador de NoCs do tipo "ciclo a ciclo", projetado com foco na acurácia dos resultados. Escrito em C++ e baseado em sua primeira versão no

livro "*Principles and Practices of Interconnection Networks*"(DALLY; TOWLES, 2004), teve sua funcionalidade expandida. A versão atual, BookSim 2.0, permite simular uma ampla variedade de topologias, incluindo *mesh* 2D, *torus* e outras específicas definidas pelo usuário (JIANG et al., 2013). Ele suporta tanto roteadores com filas de entrada quanto microarquiteturas de roteadores orientados a eventos, com suporte para canais virtuais. Os parâmetros de desempenho medidos incluem latência e taxa de transferência em função da carga oferecida. Além disso, essa abordagem permite a alteração do tamanho do *buffer*, do mecanismo de roteamento e da política de arbitragem, suportando, atualmente, apenas padrões de tráfego sintético (KHAN et al., 2017b).

Li et al. (2013) introduziram um método de roteamento baseado utilizando uma versão adaptada do simulador *BookSim*. Essa abordagem promoveu uma distribuição equilibrada do tráfego de dados, mesmo sob condições variadas de carga. Os resultados mostraram um aumento no *throughput* e uma redução na latência, especialmente em cenários de tráfego desafiadores. Esta metodologia ilustrou o potencial das novas técnicas de roteamento para melhorar a comunicação nas arquiteturas NoC (LI et al., 2013).

Kumar e Talawar (2018) introduziram um *framework* que emprega aprendizado de máquina para antecipar o desempenho das NoCs. Utilizando metodologias como regressão por vetores de suporte e redes neurais artificiais, foi possível avaliar as redes em configurações *mesh* e *torus*, alcançando uma análise expedita com um intervalo de erro de previsão entre 5% e 8% (KUMAR; TALAWAR, 2018).

Sambangi et al. (2021) desenvolveram o LPNet, um modelo que utiliza redes neurais profundas para a previsão de latência em NoC. Integrando dados de modelos analíticos com simulações do BookSim, o uso do projeto permitiu alcançar um erro de previsão abaixo de 12% para tráfegos sintéticos e específicos de aplicações. Segundo os autores, o modelo também proporcionou um avanço significativo na velocidade de análise, sendo mais de 108 vezes mais rápido que métodos convencionais. Este avanço evidenciou o potencial das técnicas baseadas em DNN na otimização e no planejamento eficiente de NoCs (SAMBANGI; MANGHNANI; CHATTOPADHYAY, 2021).

Balakrishnan et al. (2023) propuseram um *design* para um roteador projetado para detectar e responder dinamicamente ao congestionamento na rede. Esse ro-

teador monitora o tráfego em tempo real e ajusta as rotas dos pacotes para evitar áreas congestionadas, melhorando a eficiência e o desempenho da rede. No estudo, foram simuladas diversas configurações de NoC, apresentando também um mecanismo eficaz para a detecção de congestionamento. A implementação de uma estratégia de roteamento X/Y minimalista resultou em uma diminuição significativa na latência de pacotes, superando os algoritmos tradicionais e marcando um avanço importante para o desenvolvimento de NoCs mais eficientes e escaláveis (BALAKRISHNAN; VENKATESH; BHASKAR, 2023).

Reddy e Kumar (2024) introduziram uma técnica de roteamento adaptativo baseada no algoritmo Bat. Essa abordagem empregou um roteador de *pipeline* de cinco estágios para otimizar as rotas de forma adaptativa, visando aprimorar o desempenho em sistemas multiprocessadores em chip. Utilizando uma extensão do simulador BookSim, combinada com técnicas de regressão linear para avaliação de desempenho, os autores demonstraram uma melhoria na redução da latência e no aumento o *throughput* em comparação com algoritmos adaptativos tradicionais (Naresh Kumar Reddy; KUMAR, 2024).

Para concluir, Muhsen et al. (2023) apresentaram um modelo preditivo em *Artificial Neural Networks* (ANN) com otimização heurística. Utilizando o simulador Noxim, o estudo propôs uma metodologia eficiente para prever algoritmos de roteamento, facilitando uma configuração mais ágil dos sistemas MPSoC. Segundo os autores, o modelo híbrido GCAOA-ANN superou modelos anteriores, destacando-se na previsão de algoritmos de roteamento para ambientes NoC e ressaltando a importância das abordagens preditivas na otimização de sistemas em *chip* (MUHSEN et al., 2023).

3.4 Simulador Nirgam

O Nirgam é um simulador de NoC de código aberto em *SystemC*, desenvolvido em colaboração entre a University of Southampton, UK, e o Malaviya National Institute of Technology, Índia. Essa ferramenta é do tipo de eventos discretos, com precisão "ciclo a ciclo", suportando topologias *mesh* 2D e *torus*. O mecanismo de comutação adotado é o *wormhole*. Esse simulador permite ajustes como o número de canais virtuais, tamanho do *buffer* e frequência de *clock*, e suporta

mecanismos de roteamento como *source*, XY e OE. As opções de tráfego incluem geradores sintéticos que podem ser de taxa de *bits* constante, *bursty* ou baseados em rastreamento de entrada. Os parâmetros de desempenho medidos incluem a latência média por pacote, a latência média por *flit* e o *throughput* média (KHAN et al., 2017b).

Zhang et al. (2009) utilizaram o simulador Nirgam para comparar os algoritmos de roteamento XY e *Odd-Even* em uma topologia *mesh* 3x3. Os resultados mostraram que o *Odd-Even* é mais eficiente que o XY, com um parâmetro P de 1.09 contra 0.86 do XY em condições de tráfego de taxa de *bits* constante, sublinhando a superioridade do *Odd-Even* em gerenciar eficientemente o tráfego em NoCs (ZHANG et al., 2009b).

Hao et al. (2011) exploraram o equilíbrio entre o tempo de atraso e o *throughput* em NoCs utilizando o simulador Nirgam em uma topologia *mesh* 4x4. O estudo destacou a eficiência do algoritmo *Odd-Even*, que mostrou uma relação entre taxa de transferência e atraso de pacotes de 2.5358, superior aos 2.1126 alcançados pelo algoritmo XY. Segundo os autores, esses achados evidenciam a capacidade do *Odd-Even* de otimizar o desempenho em configurações NoC mais complexas (HAO et al., 2011).

Yadav et al. (2014) introduziram um modelo inovador de injeção de falhas, que abrange tanto falhas permanentes quanto transitórias, por meio da ampliação do simulador Nirgam. Essa metodologia permitiu uma análise detalhada do comportamento da rede diante de diversos cenários de falhas, concentrando-se em métricas como latência e taxa de transferência (YADAV et al., 2014).

Minzheng et al. (2016) propuseram um método de roteamento tolerante a falhas centrado em *clustering*, com foco na manutenção de baixa latência. Através do uso do simulador Nirgam, foi demonstrado que tal método consegue preservar a operacionalidade do sistema mesmo na presença de falhas inesperadas, sem impactar negativamente o desempenho global do sistema (MINZHENG; YUNZHONG; FANGFA, 2016).

Umapathy et al. (2018) desenvolveram o algoritmo de roteamento *Encircle Routing* (ER), visando otimizar a distribuição de tráfego em NoCs. Testado com o simulador Nirgam, o ER demonstrou ser superior em promover uma distribuição equilibrada de tráfego pela rede, resultando em melhorias consideráveis em latência

e congestionamento em comparação com os algoritmos XY e OE (UMAPATHY; SHAH; WANG, 2018).

Trik et al. (2022) apresentaram uma estratégia de seleção híbrida que utiliza análise de tráfego para melhorar o desempenho em NoCs. Avaliada com o simulador Nirgam, essa abordagem conseguiu diminuir o tempo de atraso dos pacotes em 38%, aumentar o *throughput* em 20% e otimizar o consumo de energia (TRIK et al., 2022).

3.5 Simulador Nostrum

O simulador *Nostrum*, desenvolvido no Royal Institute of Technology (KTH), Suécia, é uma ferramenta avançada de simulação de NoC desenvolvida em *SystemC*. Suportando topologias *mesh* 2D e *torus* e adotando mecanismos de comutação *wormhole*, ele facilita o mapeamento de aplicações em nós da rede. Inclui algoritmos de roteamento como *XY* e *deflection*, sendo altamente configurável em aspectos como tamanho da rede, frequência de *clock*, profundidade do *buffer*, tamanho do *flit* e canais virtuais (LU et al., 2002; KHAN et al., 2017b).

Lu, Sander e Jantsch (2005) se concentraram no desenvolvimento de um modelo de comunicação perfeitamente síncrona para serviços de melhor esforço na plataforma Nostrum. Eles propuseram um refinamento em três etapas para manter a consistência de sincronização em ambientes assíncronos, ilustrado por um modelo de equalizador digital. Tal estratégia ressaltou a necessidade de adaptação dos modelos de comunicação para maximizar a eficiência e eficácia em sistemas NoC (LU; SANDER; JANTSCH, 2005).

Millberg et al. (2004) investigaram a garantia de largura de banda em redes temporalmente disjuntas, usando o Nostrum NoC. A pesquisa introduziu a ideia de circuitos virtuais com contêineres em *loop*, uma solução para os desafios do roteamento *hot-potato*, visando melhorar tanto a latência quanto a largura de banda. Esse método ofereceu uma perspectiva eficaz para aprimorar o desempenho de comunicação em NoCs (MILLBERG et al., 2004).

Yue et al. (2011) discutiram estratégias de otimização e reconfiguração para plataformas NoC, com suporte do simulador Nostrum O estudo sublinhou métricas chave para otimizar redes dinâmicas e aplicações multimídia, destacando a impor-

tância do Nostrum na melhoria da eficiência de plataformas NoC destinadas a esse tipo de aplicação, oferecendo resultados importantes para o desenvolvimento de sistemas mais adaptáveis e robustos (YUE et al., 2011).

Thid, Millberg e Jantsch (2003) apresentaram a criação de um simulador NoC, desenvolvido especificamente para avaliar a arquitetura Nostrum. Implementado em *SystemC*, tal simulador reconfigurável testa diversas plataformas NoC e cargas de trabalho, demonstrando grande flexibilidade. A comparação da arquitetura Nostrum com alternativas baseadas em barramento enfatizou a importância de simulações adaptáveis na avaliação de arquiteturas NoC (THID; MILLBERG; JANTSCH, 2003).

3.6 Simulador NoCMap

NoCMap é um simulador de mapeamento de código aberto para NoC escrito em C++ por Hu et al. (HU; MARCULESCU, 2003; HU; MARCULESCU, 2005). Implementa dois algoritmos de mapeamento: BB e SA, utilizando o modelo de energia por bit para calcular a energia total mínima de comunicação do NoC. O BB é usado para o posicionamento topológico de IPs na plataforma NoC, visando minimizar o consumo de energia de comunicação, com a largura de banda da conexão como restrição. Por outro lado, um método SA *ad-hoc* também foi implementado, revelando que o BB supera a técnica SA em velocidade, mantendo resultados comparáveis (KHAN et al., 2017b).

Baseando-se no modelo EPAM XY, OE e WF), que foca em mapeamento consciente de energia e desempenho com diferentes algoritmos de roteamento, um algoritmo BB eficiente é desenvolvido. Paralelamente, o SA é avaliado, demonstrando a superioridade dos algoritmos propostos em termos de eficiência dos resultados e velocidade de simulação. Observa-se ainda que o EPAM-OE é mais preciso em aplicações reais e complexas com grandes sistemas. O *ReliableNoC*, uma expansão do NoCMap, introduz um parâmetro de confiabilidade, enriquecendo o simulador (KHAN et al., 2017b).

3.7 Análise Comparativa dos Simuladores

Buscando sintetizar as principais características dos simuladores apresentados, que são de interesse para esta tese, a Tabela 2, adaptada de (KHAN et al., 2017b), apresenta uma comparação entre os recursos e características fundamentais para experimentação e análise de opções de projeto envolvendo NoCs. Cada coluna corresponde a um simulador específico, e as linhas detalham características técnicas como linguagem de implementação, topologia suportada, padrões de tráfego, mecanismos de comutação, profundidade do *buffer*, algoritmos de roteamento, parâmetros de desempenho, modelos de energia, métodos de entrada de parâmetros e estratégias de mapeamento de tarefas. A comparação desses elementos cobre de maneira abrangente a funcionalidade e versatilidade dos respectivos simuladores para fins de análise do funcionamento de uma NoC.

De modo geral, a linguagem de programação adotada, geralmente *SystemC* ou C++, define a plataforma de desenvolvimento e execução. As topologias de rede refletem as arquiteturas de NoC que podem ser estudadas. Os padrões de tráfego e os mecanismos de comutação afetam diretamente o desempenho da simulação. A profundidade do *buffer* e os algoritmos de roteamento são cruciais para a avaliação da eficiência de comunicação e da flexibilidade das políticas de tráfego. Os parâmetros de desempenho e os modelos de energia indicam a capacidade do simulador de fornecer métricas relevantes para a otimização das NoCs. Por fim, os métodos de entrada de parâmetros e as estratégias de mapeamento são essenciais para a configuração e o controle do usuário sobre a simulação. Cada simulador é, portanto, caracterizado por sua combinação única dessas especificações, permitindo aos pesquisadores selecionar a ferramenta mais alinhada com seus objetivos de estudo específicos.

Na Tabela 3, são mostrados os anos das pesquisas que foram publicadas utilizando os simuladores descritos anteriormente, destacando como cada um contribuiu para a avaliação e desenvolvimento no contexto das NoCs.

Além dos simuladores apresentados anteriormente, duas outras ferramentas foram consideradas, porém não foram incluídas nas comparações e análises deste documento: o simulador *PAT-Noxim*, desenvolvido por Norollah et al. (2018), que é um simulador em C++ e *SystemC* projetado para aprimorar as simulações de

Tabela 2 – Resumo das Características de cada Simulador

Simulador	Noctweak	Noxim	BookSim	Nirgam	Nostrum	NoCMap
Publicação	(TRAN; BAAS, 2012)	(CATANIA et al., 2015)	(JIANG et al., 2013)	(JAIN, 2007)	(LU et al., 2002)	(HU; MARCULESCU, 2005)
Linguagem	SystemC	SystemC	SystemC	SystemC	C++	C++
Topologia	2D mesh	2D mesh	wide range	2D mesh, torus	2D mesh, torus	2D mesh
Padrão de Tráfego	synthetic, embedded	synthetic	synthetic	synthetic, embedded	synthetic	synthetic, embedded
Mecanismo de Comutação	wormhole, Roshaq, bufferless, circuit switched	wormhole with virtual channel	wormhole	wormhole with virtual channel	wormhole with virtual channel	wormhole with virtual channel
Profundidade do buffer	yes	yes	yes	yes	yes	yes
Algoritmo de Roteamento	XY, NF, WF, NL, OE, lookup table	XY, NF, WF, NL, OE, lookup table, dyad, fully adaptive	all	source routing, XY, OE	XY, deflection routing	XY, OE, WF, dyad
Parâmetros de Desempenho	power/energy throughput, latency	energy, throughput, communication delay	throughput, latency	power, throughput, latency	throughput, latency, link utilization	energy, reliability
Modelo de Energia	CMOS	Orion Model	no	Orion Model	no	bit energy model
Entrada de Parâmetros	command line	command line	log file	log file	command line	command line
Mapeamento	NMAP, RANDOM	no	no	manual	manual	BB, SA

Tabela 3 – Sequência de Anos das Publicações dos Simuladores NoC

Simulador	Anos das Publicações
NoCTweak	2014, 2017, 2019, 2021, 2022, 2023
Noxim	2016, 2021, 2023
BookSim	2013, 2018, 2021, 2023, 2024
Nirgam	2009, 2011, 2014, 2016, 2018, 2022
Nostrum	2003, 2004, 2005, 2011
NoCMap	2003, 2005, 2017

consumo de energia e temperatura, baseando-se no *Noxim*, mas sem incluir algoritmos de mapeamento (NOROLLAH et al., 2018); e o *NRTBox*, criado por Abid et al. (2015), uma *toolbox* do *software Matlab Simulink*, focada exclusivamente na simulação de desempenho de roteadores em NoCs (ABID et al., 2015).

3.8 Considerações finais

Neste capítulo, foram apresentadas publicações relevantes sobre simuladores específicos para NoCs, destacando suas características e aplicações. Observou-se que, embora o NoCMap incorpore algoritmos de mapeamento como BB e SA, sua principal contribuição está na avaliação de consumo de energia, conforme indicado pela predominância de citações dos próprios autores. Além disso, houve uma diminuição nas publicações recentes de alguns simuladores, sugerindo uma transição

para ferramentas mais novas, como Noxim e NoCTweak. Notou-se também a ausência de novos algoritmos de mapeamento em simuladores evoluídos do Noxim. Essas observações indicam a maturidade das ferramentas atuais e sugerem oportunidades para pesquisas alternativas, motivando o desenvolvimento deste projeto.

Capítulo 4

Metodologia e Desenvolvimento

Neste capítulo, são apresentados os conceitos que guiaram a criação de um método de simulação para sistemas many-core baseados em NoCs, com o objetivo de dar suporte a estudos de DSE. Em seguida, será detalhado o simulador desenvolvido para validar e analisar sua adequação aos objetivos propostos, constituindo o núcleo central dos materiais e métodos empregados nesta tese.

4.1 Visão Geral

Conforme descrito no Capítulo 1, o principal objetivo desta pesquisa foi conceber e desenvolver um método de simulação em alto nível para arquiteturas *many-core* baseadas em NoCs, denominado *Simulador NoC*, que auxilia nas atividades de DSE. A abordagem adotada visa reduzir a complexidade do aprendizado para os usuários, superando as dificuldades encontradas em outras ferramentas descritas na literatura.

Os seguintes pontos foram fundamentais para a concepção do método que originou o simulador descrito neste capítulo. Em primeiro lugar, o método visa facilitar a exploração da topologia *mesh* para NoCs. Além disso, busca simplificar a experimentação de aplicações representadas por DAGs de complexidade arbitrária.

Outro objetivo é permitir a experimentação com diversos algoritmos de mapeamento, proporcionando uma plataforma versátil para testes. Da mesma forma, o método facilita a experimentação com diferentes algoritmos de roteamento. Finalmente, procura produzir resultados numéricos que permitam inferir conclusões similares às obtidas com ferramentas mais especializadas, mas que são de uso mais complexo e limitado.

A partir desses princípios, foi desenvolvido um método de simulação para NoCs, resultando no Simulador NoC, cujo diagrama de blocos da estrutura geral é apresentado na Figura 4.1. Conforme ilustrado, o simulador é constituído por dois blocos principais: *a)* a representação da arquitetura NoC e *b)* um repositório de algoritmos que podem ser customizados para o mapeamento de tarefas e os de roteamento para *mesh* 2D. Esses dois blocos são interconectados por uma interface dedicada a converter e padronizar dados de simulação e funcionamento do simulador, garantindo que ambos os blocos possam interpretar e utilizar informações de forma unificada e coerente.

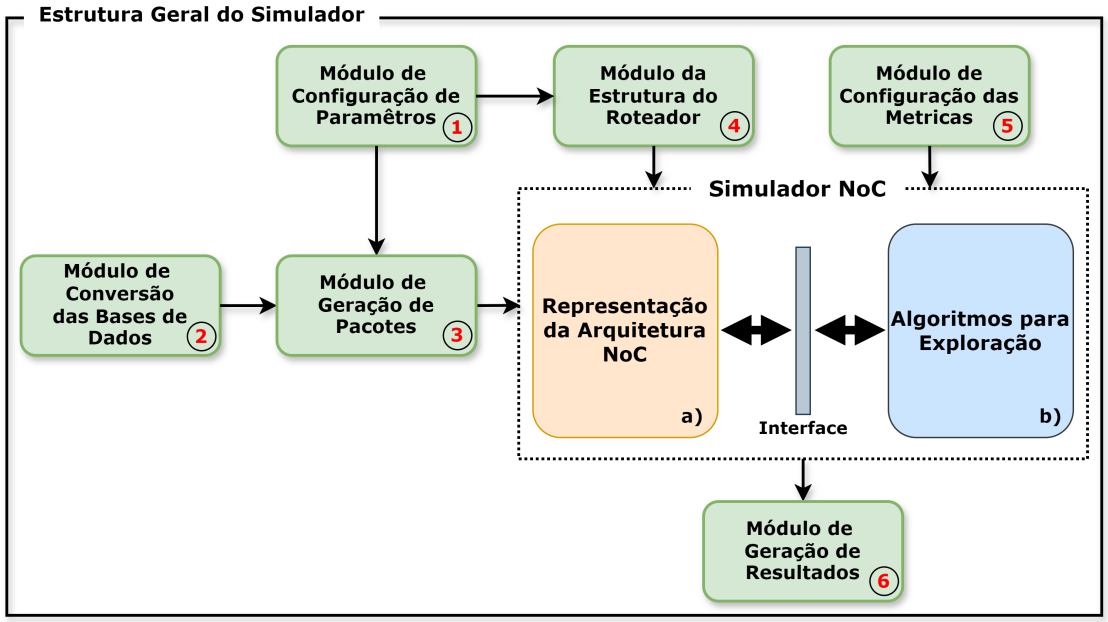

Figura 4.1 – Visão Geral do Simulador NoC.

A estrutura completa do ambiente de simulação segue uma organização modular, composta pelos módulos de apoio, numerados de 1 a 6 na Figura 4.1: Módulo

de Configuração de Parâmetros (subseção 4.3.1), Módulo de Conversão das Bases de Dados (subseção 4.3.2), Módulo de Geração de Pacotes (subseção 4.3.3), Módulo da Estrutura do Roteador (subseção 4.3.4), Módulo de Configuração das Métricas (subseção 4.3.5) e Módulo de Geração de Resultados (subseção 4.3.6).

4.2 Simulador NoC

O bloco principal do ambiente de simulação consiste em uma "representação abstrata da NoC", uma estrutura que recebe e executa simulações com base nos dados fornecidos pelos módulos de apoio (descritos mais adiante). Esta configuração estabelece o ambiente de simulação essencial para a investigação de algoritmos de mapeamento de aplicações e roteamento de pacotes. Esses algoritmos, disponíveis ou passíveis de inclusão no componente "Algoritmos para Exploração", mantêm uma interação contínua com a abstração da NoC. A interface correspondente, responsável por ajustar parâmetros e configurações específicas, pode demandar ajustes manuais para garantir a precisão dos dados, não sendo completamente automatizada. Esta estrutura é ilustrada na Figura 4.2, e detalhada nas próximas seções deste capítulo.

4.2.1 Representação da Arquitetura NoC

O principal componente do ambiente de simulação é a "representação abstrata da NoC", uma estrutura que recebe e executa simulações com base nos dados fornecidos pelos módulos de apoio. Tal configuração estabelece o ambiente necessário para investigar algoritmos de mapeamento de aplicações e roteamento de pacotes. Esses algoritmos, disponíveis ou passíveis de inclusão no componente "Algoritmos para Exploração", interagem continuamente com a abstração da NoC. A interface responsável por ajustar parâmetros e configurações específicas pode exigir ajustes manuais para garantir a precisão dos dados, não sendo completamente automatizada. Essa estrutura é ilustrada na Figura 4.2 e detalhada nas próximas seções deste capítulo.

Tal representação é a base necessária para a simulação em um ambiente NoC controlado, criando o cenário onde os "Algoritmos para Exploração"são testados e

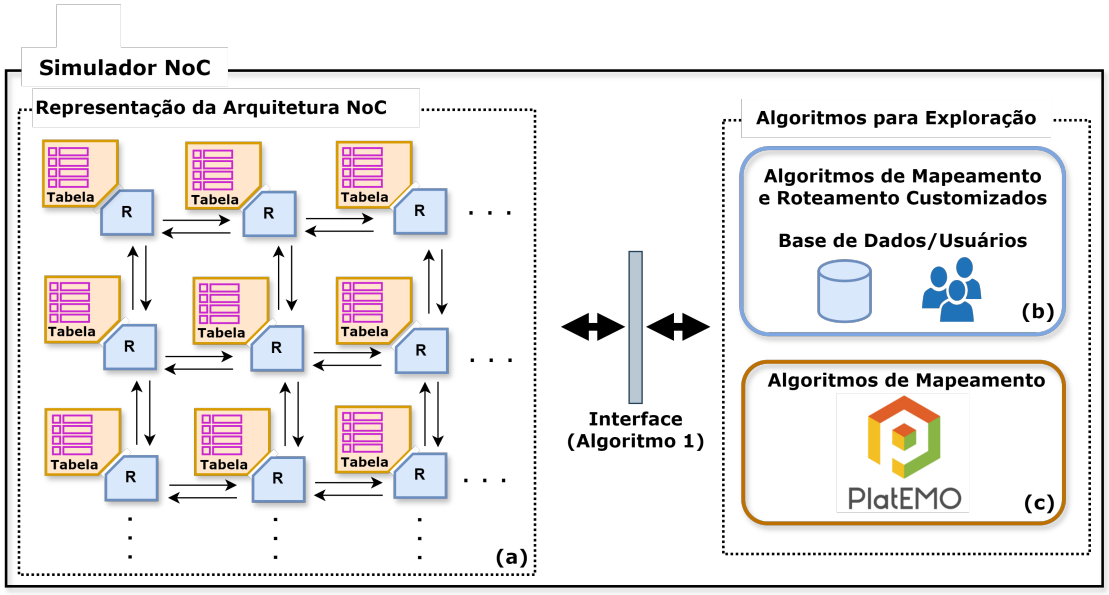

Figura 4.2 – Representação do Simulador de Alto Nível para NoC.

validados. Os pacotes (representações abstratas da largura de banda que podem percorrer pelos *links*) são injetados na rede com base nos parâmetros de simulação e roteados conforme os algoritmos implementados. O objetivo é avaliar aspectos críticos como a eficiência do roteamento e o consumo de energia, fundamentais para o *design* e a operação de sistemas NoC.

O uso do termo "Interface"enfatiza a função essencial de mediação entre diferentes componentes do sistema, facilitando interações e a execução de algoritmos. Ela gerencia desde a inicialização da simulação até o roteamento de pacotes conforme os algoritmos selecionados. Coordena a entrega de pacotes, as rotinas de roteamento e a atualização dos estados dos roteadores e pacotes, unificando as funções operacionais durante a simulação. Essa abordagem direciona os usuários a focarem na eficiência da NoC e na exploração dos algoritmos para otimização, simplificando a pesquisa e o desenvolvimento ao reduzir a complexidade inerente.

No pseudocódigo no Algoritmo 1, é descrito resumidamente o núcleo dessa funcionalidade, detalhando como a interface administra a movimentação e entrega de pacotes dentro da NoC. Ele apresenta as etapas de processamento de pacotes desde a origem até o destino, enfatizando a gestão eficiente do tráfego e do consumo energético. A simplicidade do pseudocódigo destaca a capacidade da in-

Algoritmo 1: Pseudocódigo Simplificado da Interface de Gerenciamento

Entrada: Estado inicial dos roteadores R, Posição do roteador iG, Dimensões da NoC nR, nC, Tempo de simulação iTS
Saída : Estado atualizado dos roteadores R

```
1  Função Simular(R, iG, nR, nC, iTS):
2      energia ← vetor de zeros;
3      para i ← 1 até tamanho de R faça
4          pacotes ← obterPacotesNoBuffer(R[i], iTS);
5          para cada pacote em pacotes faça
6              se destinoDoPacote(pacote) é iG então
7                  entregarPacote(R[i], pacote);
8                  atualizarEnergia(R[i], pacote);
9              fim
10             senão
11                 proximoRoteador ← determinarProximoRoteador(R[i],
                     pacote, nR, nC);
12                 encaminharPacote(proximoRoteador, pacote);
13                 atualizarEnergia(R[i], pacote);
14             fim
15         fim
16     fim
17     return R;
```

terface de abstrair complexidades, permitindo uma interação intuitiva e eficaz com a simulação.

4.2.2 Repositório de Algoritmos para Exploração

Esse componente do ambiente de simulação, ilustrado na Figura 4.2b, representa um diferencial ao integrar-se com a abstração da estrutura da NoC (Figura 4.2a). Sua estrutura permite que os usuários utilizem algoritmos de mapeamento e roteamento já implementados (algoritmos de otimização que foram adaptados ao contexto de mapeamento, assim como os de roteamento para *mesh* 2D), ou possam customizar/desenvolver e inserir suas próprias abordagens. Essa flexibilidade cria um ambiente de simulação ajustável e controlado para o desenvolvimento e testes desses algoritmos e de outros parâmetros da estrutura NoC. Além disso, a capacidade de realizar comparações diretas entre as abordagens dos usuários e as

consolidadas na literatura é significativamente ampliada, oferecendo um método simplificado para a análise e validação da eficácia dos algoritmos desenvolvidos ou adaptados.

Como exemplo, a seguir é descrito o algoritmo de busca tabu (OLIVEIRA; CARVALHO; KREUTZ, 2021), incluído no repositório de algoritmos customizados (Algoritmo 2). O mesmo integra as funções específicas do simulador e seus respectivos módulos, coordenando o processo de otimização. O procedimento inicia-se com a definição dos parâmetros de busca, continua com a execução da busca propriamente dita e conclui com a coleta e análise dos resultados obtidos. As funções incluídas são abrangentes, cobrindo todas as etapas necessárias para a realização da simulação, avaliação e otimização dentro do ambiente definido.

Algoritmo 2: Pseudocódigo Simplificado do Algoritmo de Busca Tabu.

Entrada: Tamanhos dos grafos *graphSizes*, Número de iterações *numIterations*
Saída : Melhores mapeamentos *bestMappings* para cada tamanho de grafo

```
1 para cada tamanho em graphSizes faça
2 |   grafo ← CarregarGrafo(tamanho);
3 |   diretorio ← VerificarCriarDiretorio(tamanho);
4 |   bestMappings ← vetor de numResultsPerGraph posições;
5 |   para j ← 1 até numResultsPerGraph faça
6 |   |   bestMappings[j] ← BuscaTabu(grafo, numIterations);
7 |   fim
8 |   Salvar bestMappings em diretorio;
9 fim
```

A execução desse algoritmo é baseada em uma série de grafos com diferentes números de tarefas, centralizando as funções específicas do simulador e do algoritmo no mesmo local. Iniciando com o carregamento do grafo, o algoritmo procede para verificar ou criar um diretório para armazenar os resultados. Com um vetor preparado para registrar os melhores mapeamentos, a busca tabu é executada, e ao concluir o número estabelecido de iterações, os resultados são salvos no diretório correspondente. Esse fluxo operacional não apenas facilita a comparação e análise de desempenho em vários cenários estruturais de grafos, mas também unifica as

operações de otimização em um ambiente integrado, otimizando a interação entre as diferentes funcionalidades do simulador e as técnicas de otimização aplicadas. Esses resultados, então, são submetidos ao módulo de resultados para gerar o valor referente à métrica escolhida.

Com relação aos algoritmos de roteamento presentes na base de dados ou implementados pelos usuários, é preciso definir um algoritmo de mapeamento pré-determinado como referência. Isso estabelece uma base consistente para a avaliação comparativa das variações e das escolhas estratégicas de roteamento. Ao fixar um algoritmo de mapeamento específico, torna-se possível realizar modificações controladas e selecionar estratégias de roteamento para testes e análises detalhadas. Tal abordagem facilita uma avaliação focada na eficiência e eficácia das diversas estratégias de roteamento, levando em consideração o impacto direto dessas escolhas no desempenho geral do sistema simulado. Ademais, permite usar uma métrica simplificada de consumo de energia como base para a comparação.

4.2.3 Repositório de Algoritmos PlatEMO

Esse componente consiste na integração do ambiente de simulação do simulador NoC com a plataforma PlatEMO, conforme ilustrado na Figura 4.2c, ampliando significativamente as possibilidades de simulação ao oferecer uma vasta gama de algoritmos evolutivos adaptáveis aos desafios de otimização em NoCs. A escolha dessa plataforma é justificada pela sua capacidade de apoiar estratégias de otimização tanto para múltiplos objetivos quanto para um único objetivo, sendo particularmente útil para uso em DSE. Sua arquitetura de código aberto permite que os usuários acessem e adaptem uma ampla variedade de algoritmos de otimização, disponíveis publicamente, aos problemas específicos enfrentados no simulador. Além disso, a constante atualização da plataforma com novos algoritmos personalizados, disponibilizados no GitHub (TIAN et al., 2017), incrementa a adaptabilidade do simulador. Tal flexibilidade é essencial para o gerenciamento eficiente de tarefas de otimização complexas, satisfazendo as demandas de aplicações que requerem soluções imediatas e personalizadas.

Na Figura 4.3, é exibido o diagrama de fluxo do processo de otimização executado pela plataforma PlatEMO. O algoritmo é inicializado através do arquivo

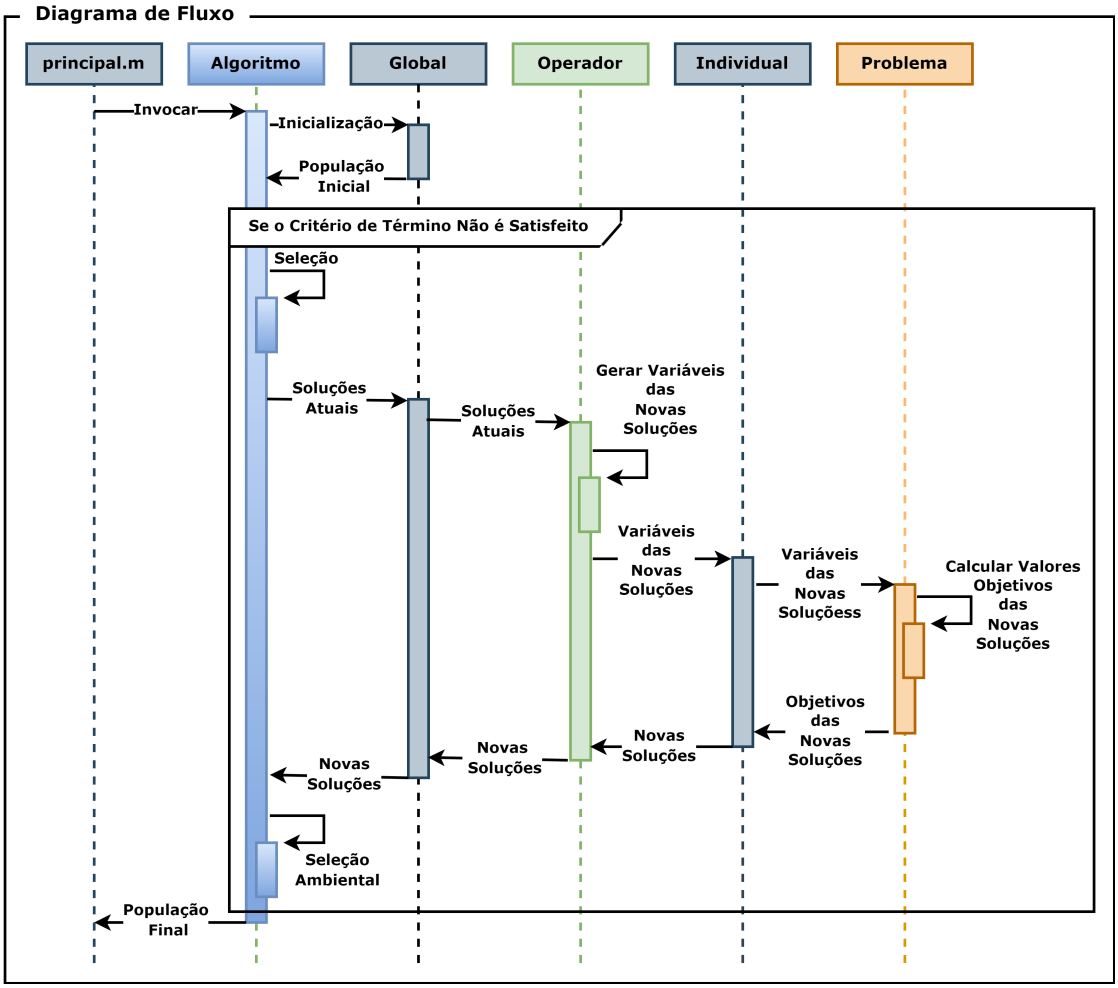

Figura 4.3 – Diagrama de Fluxo do Processo de Otimização no PlatEMO. (Adaptado de (TIAN et al., 2017))

`principal.m`, que desencadeia a formação de uma população inicial composta por um conjunto de soluções potenciais. O fluxo de otimização se desdobra com a seleção de soluções baseada em determinados critérios de desempenho. Caso os critérios de conclusão do processo, como a satisfação de uma condição específica, a exaustão de um número predefinido de iterações ou a obtenção de um limiar de qualidade da solução, não sejam cumpridos, o fluxo avança para a geração de novas soluções. Nessa fase, operações de busca e refinamento são aplicadas para explorar o espaço de soluções e gerar novas propostas, que posteriormente passam por uma avaliação onde são mensurados os valores das funções-objetivo.

Após a avaliação, dependendo da configuração do algoritmo de otimização, pode ocorrer uma nova rodada de seleção com as soluções atualizadas ou prosseguir para uma seleção ambiental, que determina as soluções que avançarão para a próxima geração. O ciclo iterativo prossegue até que o critério de conclusão seja atingido. A iteração final resulta na população final, composta pelas soluções que se mostraram mais adequadas após sucessivas gerações de busca e melhoria.

Algoritmo 3: Pseudocódigo de Otimização Evolutiva - PlatEMO.

```
Entrada: Grafo do problema graph, Número de indivíduos nInd
Saída  : População final PopFinal
Função MNoc_G1(graph, nInd):
    Definir objetivos e dimensões do problema;
    Inicializar a população com base em graph e nInd;
    while condição de término não atendida do
        Selecionar soluções atuais e gerar novas;
        Avaliar novas soluções e atualizar a população;
    end
    Aplicar seleção ambiental e retornar a população final;
```

O pseudocódigo, descrito no Algoritmo 3, detalha a lógica implementada no arquivo denominado "MNoc_G1", que orienta as operações para otimizar um determinado grafo de tarefas. O processo começa com a inicialização de uma população de soluções, seguida por uma seleção baseada em desempenho. A seguir, ocorre a geração e avaliação de novas soluções, um procedimento que se repete em ciclos até alcançar um conjunto de soluções aprimoradas. Esse arquivo é compatível tanto com *Command-line Interface* (CLI) quanto com a *Graphical User Interface* (GUI) da ferramenta, o que promove uma utilização interativa e flexível, incluindo a geração de tabelas em Excel e LaTeX dos resultados.

Para que a integração do simulador do projeto à plataforma PlatEMO seja efetiva, é necessário ajustar os códigos-fonte, enfatizando o arquivo "MNoc_G1", para definir os parâmetros do problema de otimização, incluindo a criação da população inicial e a avaliação dos objetivos com base nos dados inseridos. Com esses ajustes, o simulador é reconhecido pela PlatEMO como um "problema", permitindo a aplicação dos algoritmos de otimização da plataforma nos desafios específicos do simulador proposto.

Essa estratégia destaca a abordagem evolutiva e a precisão na avaliação, atributos essenciais do processo de otimização da PlatEMO, agora integrados ao ambiente de simulação do projeto proposto. As funções centrais, que constituem a estrutura dos módulos do simulador, são organizadas na pasta "Problems", dentro de um subdiretório denominado "simulador"(ilustrado no diagrama da Figura 4.3 pelo retângulo "Problema"). Tal organização simplifica a identificação e a inserção de problemas de otimização pelo simulador no contexto da plataforma, otimizando a interação dentro do ambiente de simulação. Isso inclui a aplicação desses algoritmos na avaliação das estratégias de roteamento implementadas no simulador.

4.3 Módulos de Apoio ao Simulador NoC

Nesta seção, é apresentada a descrição detalhada da funcionalidade e importância de cada módulo de apoio individual, abordando as especificações, os processos e as contribuições de cada um desses componentes para o funcionamento geral do ambiente de simulação. Ao se explorarem esses aspectos específicos, enfatiza-se como cada módulo facilita uma análise eficaz e detalhada das NoCs. Com isso, busca-se o alinhamento com o objetivo de suavizar a curva de aprendizado e permitir aos usuários uma assimilação mais simplificada dos conceitos relacionados a essa categoria de redes, em um ambiente que ofereça maior controle e clareza das etapas. Para esclarecimento, dentro deste projeto, o termo "nó"ou "nó computacional"refere-se à integração de um PE com uma IP, representando uma unidade abstrata com capacidades de processamento e comunicação. Além disso, os módulos são definidos como funções de programação específicas ou um conjunto delas, que juntos formam o núcleo ou a estrutura do projeto proposto.

4.3.1 Módulo de Configuração de Parâmetros

A configuração de parâmetros, ilustrada na Figura 4.11, atua como um conjunto inicial de definições que orienta o ambiente de simulação. Eles são adaptáveis, permitindo tanto especificações direcionadas quanto configurações de uso mais amplo, adequando-se às particularidades de cada algoritmo de mapeamento ou roteamento adotado. Por exemplo, eles se aplicam a estratégias que vão desde

algoritmos evolutivos e com características genéticas até métodos de busca local, entre outros. Ou seja, eles têm influência direta em aspectos críticos da simulação, como os padrões de tráfego derivados dos grafos de aplicações e as dimensões do *grid*, que delimitam a topologia física da rede, são eles:

Nº de Linhas (nR): Especifica o total de linhas para a configuração da topologia *mesh* 2D, refletindo a dimensão vertical da NoC;

Nº de Colunas (nC): Determina o total de colunas para a configuração da topologia *mesh* 2D, refletindo a dimensão horizontal da NoC;

Dimensão do Grid (gridSize): Determinada pela multiplicação do número de linhas (nR) pelo número de colunas (nC), indicando o total de nós computacionais na topologia *mesh* 2D;

Nº de Soluções (nSol): É o equivalente ao número de indivíduos (nInd) que vão formar a população, em cada algoritmo com suas características;

Custo (metrics): Define o custo com base na métrica escolhida para avaliar o desempenho das configurações de NoC, auxiliando na identificação das soluções mais eficazes. Tal métrica, seja proveniente da implementação original do simulador ou adicionada pelo usuário, desempenha um papel essencial na otimização e na comparação de resultados;

Nº de Evoluções (nEval): Representa o número total de iterações ou avaliações da função objetivo realizadas pelo algoritmo;

Nº de Execuções (nRuns): Indica quantas vezes o algoritmo é executado independentemente, com cada execução partindo de condições iniciais distintas.

É importante destacar que alguns algoritmos podem requerer a inclusão direta de parâmetros inseridos em partes distintas do seu código, dependendo das especificidades de sua implementação.

4.3.2 Módulo de Conversão das Bases de Dados

Esse módulo, representado na Figura 4.12, é responsável pela conversão dos dados de entrada para o formato padrão utilizado no ambiente de simulação. Eles podem ser destacados como grafos de aplicações, uma categoria que pode incluir tanto DAGs quanto *Benchmarks*. Essa classificação reflete a versatilidade dessa

abordagem em capturar diferentes aspectos e necessidades de simulação em NoCs, seja representando processos computacionais arbitrários ou cenários de uso reais.

4.3.2.1 Grafos de Aplicações

Os grafos de aplicações, essenciais na modelagem de NoCs, proporcionam um mapeamento detalhado das interdependências e coordenam o fluxo de tarefas. Ao visualizar as sequências de execução e as comunicações inter-tarefas, viabilizam a simulação e otimização das arquiteturas da rede. Essa categoria se destaca por sua capacidade de detalhar a interação entre tarefas, sendo importante para alinhar o *design* teórico com as demandas práticas, promovendo o desenvolvimento avançado de sistemas *many-core* eficientes. Seguindo nesse contexto, serão abordados especificamente os DAGs e *Benchmarks*, que são elementos integrantes deste cenário e contribuem significativamente para a análise e avaliação dos problemas que envolvem esse contexto (GROSS; YELLEN; ANDERSON, 2018; JAIN, 1991).

Prosseguindo com essa análise, o modelo de aplicação emprega grafos gerados pela ferramenta TGFF (DICK, 2022), que, através de parâmetros estocásticos, representam cargas de trabalho teóricas ou arbitrárias. Essa abordagem não só complementa a simulação e otimização discutidas anteriormente, como também introduz uma camada adicional de precisão ao detalhar o funcionamento interno de sistemas embarcados.

Formalmente, um DAG é expresso por $G(V, E, Q, W)$, onde $v_i \in V$ simboliza as tarefas individuais dentro do grafo, representando as unidades de trabalho que necessitam ser executadas. Ademais, $e_{i,j} \in E$ descreve as dependências direcionais entre essas tarefas, estabelecendo uma sequência de execução com base nas relações de precedência. Os conjuntos Q e W definem, respectivamente, qualidades e pesos atribuídos às tarefas e às arestas, detalhando aspectos específicos desses elementos que podem variar conforme a aplicação (FANG; YU; WEI, 2020).

Adicionalmente, os *benchmarks* industriais para sistemas embarcados, como 80211arx, mms, e3s_autoindust_ori, mwd, mpeg4, cavlc, e3s_consumer_ori, vce, vopd, wifirx e e3s_telecom_ori, refletem cenários de uso real e são integrados às bases de dados de alguns dos simuladores discutidos anteriormente. O número de tarefas empregadas por esses *benchmarks* varia entre 12 e 30 (JAIN, 1991).

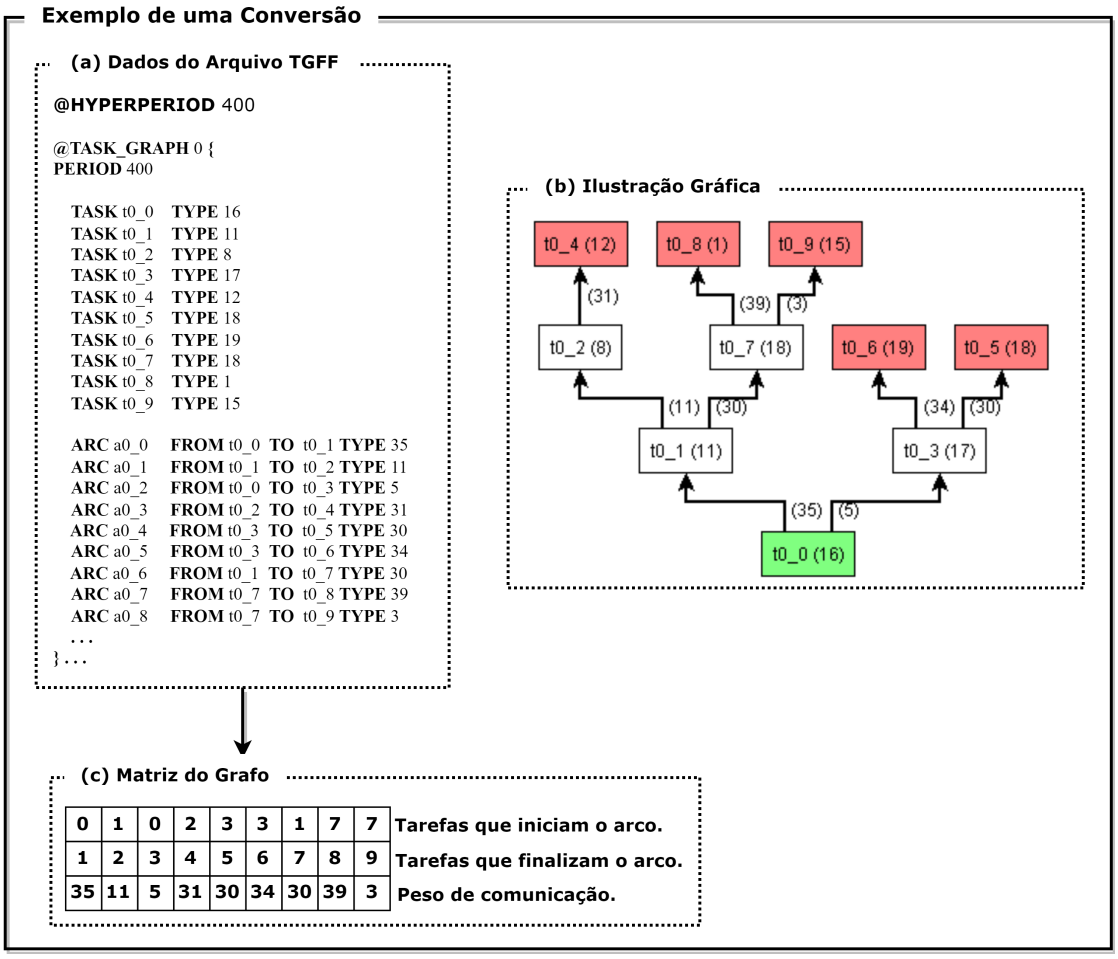

Figura 4.4 – Exemplo da Conversão do Arquivo da Aplicação para a Matriz de Dados.

Na Figura 4.4, são apresentadas ilustrações dos componentes necessários para a conversão de um grafo de tarefas para a matriz do grafo, aceita na estrutura do simulador de alto nível. A Figura 4.4a detalha um arquivo TGFF que descreve as tarefas individuais e seus tipos, além das relações de dependência entre elas (arcos). A Figura 4.4b representa a transformação da descrição textual em uma representação visual, em que cada nó computacional simboliza uma tarefa e as setas indicam as dependências, com números que denotam os tipos das tarefas e os pesos de comunicação dos arcos.

Por fim, na Figura 4.4c, essas informações são sintetizadas em uma estrutura matricial, com as duas primeiras linhas representando, respectivamente, as tarefas

de origem e destino dos arcos, e a terceira mostrando o peso de comunicação entre elas. As reticências indicam que apenas uma parte do arquivo é apresentada, sugerindo que há mais conteúdo além do mostrado.

4.3.3 Módulo de Geração da Abstração dos Pacotes

A geração da abstração dos pacotes, ilustrada na Figura 4.13, é orientada pela matriz do grafo (Figura 4.4), construída a partir dos dados extraídos do grafo de aplicações. O procedimento envolve identificar as tarefas que demarcam os arcos no grafo, estabelecendo as coordenadas (x,y) para o início e o término da comunicação nos nós computacionais. Para cada pacote gerado, o momento exato da criação e os custos de comunicação são calculados e registrados. Esse método assegura que a geração dos pacotes espelhe com precisão as dependências e prioridades definidas no grafo de aplicações.

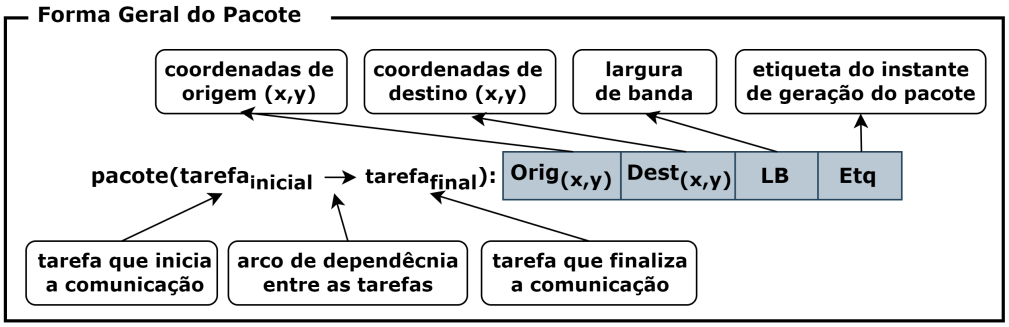

Figura 4.5 – Forma Geral do Pacote Utilizado pelo Simulador Proposto.

Na ilustração da Figura 4.5, o pacote é retratado como uma estrutura de dados para a comunicação na rede. Ele é definido pelas coordenadas de origem e destino (x, y), que determinam os nós computacionais de emissão e recepção do pacote, respectivamente. A largura de banda (LB) associada indica de forma abstrata a quantidade de dados a serem transmitidos no referido *link*.

A etiqueta (*Etq*) do pacote registra o momento de sua geração, sendo essencial para o monitoramento da rede e facilitando a análise de métricas pertinentes e o rastreamento da ordem de transmissão dos pacotes. O pacote, portanto, personifica a transferência de informação e as relações de dependência entre diferentes tarefas.

O pseudocódigo do processo de geração de pacotes abstratos, descrito no Algoritmo 4, inicia com a criação de uma estrutura de dados para armazenar as informações para o encaminhamento de pacotes na abstração de alto nível da NoC. Ela é inicializada com zeros e preparada para conter as coordenadas de origem (Sx, Sy) e destino (Dx, Dy) do pacote em questão, além de parâmetros específicos da transmissão, como a largura de banda e a etiqueta do instante de geração.

Algoritmo 4: Pseudocódigo do Processo de Geração de Pacotes

Entrada: Estrutura de roteadores R, grafo, mapeamento *map*, número de linhas nR, número de colunas nC
Saída : Estrutura de roteadores atualizada R

```
1  Função GeracaoDePacote(R, grafo, map, nR, nC):
2      para cada arco iGraph no grafo faça
3          Inicialize o pacote pkt com zeros;
4          para cada posição iSrc de 1 até nR × nC faça
5              Verifique se map(iSrc) é igual ao nó de origem no arco iGraph;
6              Se verdadeiro, defina idxSrc como iSrc;
7          fim
8          Converta idxSrc para coordenadas (Sx, Sy) e atualize pkt;
9          para cada posição iDest de 1 até nR × nC faça
10             Verifique se map(iDest) é igual ao nó de destino no arco
                iGraph;
11             Se verdadeiro, defina idxDest como iDest;
12         fim
13         Converta idxDest para coordenadas (Dx, Dy) e atualize pkt;
14         Defina o peso de comunicação e outros parâmetros de pkt;
15         Se R(idxSrc).table estiver vazio, adicione pkt como a primeira
            entrada;
16         Senão, adicione pkt à próxima posição livre;
17     fim
18     Retorne R;
```

Para cada arco no grafo, que simboliza as ligações entre os nós na rede, o algoritmo procede à identificação do nó computacional de origem. Isso é realizado através da comparação entre os identificadores dos nós e os valores presentes no mapeamento do *grid*, definido pelo produto das dimensões da rede ($nR \times nC$). A identificação correta do nó de origem é importante, pois determina o ponto inicial

da transmissão do pacote.

Utilizando a função `ind2sub`, o índice do nó de origem é convertido para coordenadas bidimensionais, refletindo sua posição física dentro do *grid*. Essas coordenadas são, então, registradas no pacote, conforme ilustrado na Figura 4.5, estabelecendo o ponto de partida para o encaminhamento dele. O mapeamento inicial das tarefas aos PEs é determinado pela distribuição de cada solução do problema. Com uma população de 100 soluções, o mapeamento permanece fixo durante toda a iteração atual, mas pode ser alterado na geração da próxima população, dependendo das características dos algoritmos utilizados. Esse mapeamento pode mudar a cada tempo de simulação, correspondendo ao índice do número de soluções encontradas. Isso permite uma maior flexibilidade e adaptação do sistema, ajustando o posicionamento das tarefas de acordo com as melhores soluções de mapeamento e roteamento identificadas em cada iteração, otimizando continuamente o desempenho da NoC.

Um procedimento similar é adotado para localizar o nó de destino dentro do *grid*, garantindo que cada dependência entre as tarefas do grafo seja mapeada para um conjunto específico de coordenadas de origem e destino. A conversão do índice do nó de destino em coordenadas bidimensionais completa a definição das rotas de transmissão.

Com as coordenadas de origem e destino definidas, o pacote é complementado com informações adicionais, como o peso de comunicação, e é alocado na estrutura da tabela do nó de origem, seguindo a ordem de geração em relação à estrutura da matriz do grafo. A metodologia descrita no pseudocódigo garante uma abordagem sistemática para a simulação de transmissões de pacotes em NoCs, permitindo uma análise detalhada do desempenho da comunicação entre os nós.

4.3.4 Módulo da Estrutura Geral do Roteador

No contexto de NoC, uma das abordagens para o gerenciamento eficiente de dados é a implementação de uma estrutura de roteamento otimizada (Figura 4.14). Na Figura 4.6, é apresentada uma estrutura de dados de alto nível projetada para esse propósito, representando a organização lógica dos roteadores dentro de uma topologia *mesh*. Essa estrutura é essencial para simular o comportamento da rede

em um ambiente controlado, permitindo a análise detalhada do fluxo de pacotes e o controle de tráfego.

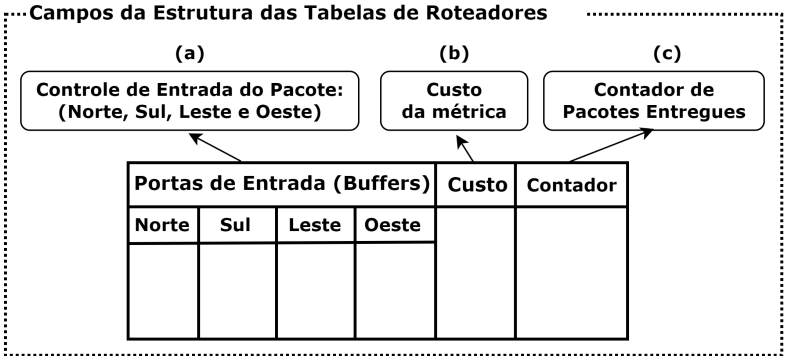

Figura 4.6 – Estrutura de Dados de Alto Nível dos Roteadores.

A estrutura detalhada incorpora vários campos essenciais para o controle e monitoramento do fluxo de pacotes dentro da estrutura de alto nível do roteador e na arquitetura NoC do simulador. Dentre esses campos, destacam-se as portas de entrada (*buffers*), indicadas na Figura 4.6a. Elas são fundamentais, atuando como pontos de recepção para a abstração dos pacotes provenientes das direções Norte, Sul, Leste e Oeste, funcionando como áreas de armazenamento temporário, preservando os dados recebidos até que sejam processados e redirecionados para a próxima etapa do roteamento.

Existem *buffers* para conter o envio de um pacote local caso haja um fluxo passando por este nó. Por padrão, a profundidade do *buffer*, que é a capacidade do *link*, é mantida no maior valor do peso de comunicação de uma dependência no *DAG*. Em cada iteração, os *buffers* são liberados, mas mantêm informações dos pacotes que ficaram para ser encaminhados na próxima iteração, seguindo a regra FIFO para o próximo encaminhamento. Isso garante que o simulador possa lidar com o fluxo contínuo de pacotes, mantendo a eficiência do roteamento e a integridade das transmissões de dados. Uma regra aplicada é que um pacote só pode ser enviado no momento seguinte ao de sua geração ou deve aguardar na sequência em que chegou à porta, seja do roteador intermediário ou de destino. A única exceção ocorre se o pacote que o precede não puder ser suportado pela largura de banda disponível no *link*.

Outro campo (custo), como mostrado na Figura 4.6b, é representado por um vetor que registra o custo associado à transmissão de cada pacote do roteador de origem ao destino, com base em várias métricas, como o custo total simplificado de energia (Subseção 4.3.5) e outras, permitindo uma análise detalhada do desempenho da rede. Por fim, o contador de pacotes entregues (contador), evidenciado na Figura 4.6c, registra o número de pacotes entregues, incrementando-se a cada entrega bem-sucedida, refletindo o volume total de tráfego de pacotes efetivamente processado pelo roteador.

Algoritmo 5: Pseudocódigo da Estrutura da Tabela dos Roteadores

Entrada: número de linhas nR, número de colunas nC, grafo
Saída : Estrutura de roteadores R

Função `EstruturaRoteador(`*nR, nC, grafo*`):`
  Inicialize a estrutura de roteadores R;
  para *cada posição iGrid de 1 até* $nR \times nC$ **faça**
    Inicialize $R(iGrid).link$ como um vetor de zeros com 5 posições;
    Defina cada posição de $R(iGrid).link$ com o valor da largura da banda;
    Inicialize $R(iGrid).table$ o campo da estrutura para controle de envio e recebimento dos pacotes, custo do cálculo métrica e contagem de pacotes;
    Defina $R(iGrid).dlv$ como 0 para armazenar o número de pacotes entregues;
    Defina $R(iGrid).metric$ como 0 para armazenar o valor da métrica associada;
  fim
  Retorne R;

O processo de inicialização de roteadores, descrito no Algoritmo 5, detalha uma abordagem sistemática para a configuração de roteadores na abstração da rede NoC do projeto, visando otimizar o roteamento de pacotes e assegurar uma comunicação eficiente entre os nós. A seguir, será apresentada uma análise das etapas desse procedimento.

Iterando sobre o *grid* dos roteadores, o índice `iGrid` é utilizado para garantir que cada unidade na rede seja configurada de forma individual. Tal estratégia assegura que a configuração seja personalizada e esteja alinhada com a localização

específica de cada roteador, fator essencial para a eficiência do roteamento e a redução de congestionamentos. Inicializando os campos dos *links* com zeros e definindo suas capacidades com base no grafo da rede, estabelecem-se os parâmetros de transmissão para cada roteador, otimizando a largura de banda e configurando os *links* entre os roteadores para suportar adequadamente o tráfego de dados.

Em sequência, a criação de um *buffer* temporário para cada roteador facilita o gerenciamento eficaz do tráfego de entrada, uma medida para evitar a perda de pacotes. Isso garante que os dados sejam armazenados temporariamente até que possam ser processados ou redirecionados. Prioriza-se o encaminhamento do tráfego local, garantindo que os pacotes gerados no nó sejam encaminhados antes dos pacotes em trânsito. A duração do envio de pacotes é emulada levando em consideração os tamanhos dos pacotes e as taxas de transmissão na simulação. Se o próximo nó (*hop*) estiver em uso e o fluxo for armazenado em *buffer*, os tempos de espera são contabilizados. Isso inclui a duração do tempo em que os pacotes aguardam nos *buffers* e o tempo efetivo de transmissão, garantindo que todos os detalhes temporais sejam levados em conta para uma simulação precisa. O emprego de uma variável para contabilizar os pacotes entregues com sucesso habilita o monitoramento do desempenho do roteador, permitindo sua utilização posterior em cálculos de métricas ou para análise de conceitos relacionados.

A divisão da estrutura de cada roteador, conforme descrita, não apenas facilita a manutenção e expansão da rede, mas também estabelece uma gestão eficiente do tráfego. Ao configurar cada roteador para otimizar o roteamento de pacotes e garantir a comunicação eficaz, o procedimento descrito promove uma rede mais resiliente, confiável e capaz de se adaptar a diferentes volumes de tráfego e requisitos de comunicação.

4.3.5 Módulo de Configuração das Métricas

No desenvolvimento deste projeto de doutorado, esta subseção descreve a metodologia adotada para a configuração da métrica desejada pelo usuário (Figura 4.15). Especificamente, para os experimentos e análises no contexto do projeto, foi adotada a métrica de **custo simplificado de energia**. Contudo, ressalta-se a possibilidade de implementar e inserir outras métricas no contexto de simula-

ção, visando à estimativa de parâmetros de desempenho em sistemas *many-core* com arquiteturas NoC. Destaca-se, assim, a importância da abstração utilizada não só para facilitar a assimilação dos conceitos, mas também para a análise de desempenho da rede.

Este módulo detalha a aplicação da métrica de custo de energia simplificado, utilizada para avaliar o consumo energético em sistemas *many-core* com arquiteturas NoC. A métrica, conforme descrita pela Equação 7, facilita a estimativa do consumo energético ao considerar a quantidade de *links* que cada pacote atravessa e o custo energético inicial por *link*, que é o peso de comunicação atribuído ao pacote de acordo com o grafo utilizado. Tal metodologia destaca como as estratégias de roteamento influenciam diretamente o consumo de energia, além de reforçar o compromisso do projeto com a oferta de uma fundamentação para análises comparativas e futuras otimizações.

$$C_{\text{total}} = \sum_{i=1}^{n} (q_i \cdot C_{\text{inicial}}) \tag{7}$$

Onde:

a) C_{total} representa o custo simplificado total de energia;

b) n é o número total de pacotes;

c) q_i indica o número de *links* atravessados pelo pacote i;

d) C_{inicial} é o custo simplificado de energia inicial atribuído por *link*.

A Lei de Joule, que estabelece que o calor gerado em um condutor é proporcional ao quadrado da corrente elétrica, à resistência do condutor e ao tempo (HALLIDAY; RESNICK; WALKER, 2023), serve como base teórica para compreender o consumo energético em sistemas *many-core* com arquiteturas NoC. Analogamente, a Equação 7 simplifica essa relação ao estimar o consumo energético com base no número de *links* que cada pacote atravessa e no custo inicial atribuído a cada *link*.

Essa simplificação, embora não incorpore diretamente todas as variáveis da Lei de Joule, captura a essência do princípio de que o consumo energético aumenta com a quantidade de atividade no sistema (nesse caso, o tráfego de pacotes), oferecendo uma abordagem prática para a análise de desempenho energético em arquiteturas NoC. Os pseudocódigos fornecidos complementam essa equação, proporcionando

um método para calcular o número de *links* atravessados, o que, por sua vez, permite uma estimativa direta do custo energético. Isso reitera a importância de estratégias de roteamento eficientes para a minimização do consumo de energia.

Algoritmo 6: Pseudocódigo para Determinar as Coordenadas do Roteadores

Entrada: Porta de saída do roteador de origem $outUS$, posição X Sx, posição Y Sy, número de linhas nR, número de colunas nC
Saída : Porta de entrada do roteador de destino $inDS$, índice do roteador $iRDS$

```
1  Função Linked(outUS, Sx, Sy, nR, nC):
2      Defina aux como um vetor contendo nR e nC;
3      switch outUS do
4          case 1 do
5              Defina inDS como 2 ;                          // Sul
6              Decrementa Sx por 1;
7              iRDS ← calcule o índice linear (Sy, Sx) usando aux;
8          end
9          case 2 do
10             Defina inDS como 1 ;                          // Norte
11             Incremente Sx por 1;
12             iRDS ← calcule o índice linear (Sy, Sx) usando aux;
13         end
14         case 3 do
15             Defina inDS como 4 ;                          // Oeste
16             Incremente Sy por 1;
17             iRDS ← calcule o índice linear (Sy, Sx) usando aux;
18         end
19         case 4 do
20             Defina inDS como 3 ;                          // Leste
21             Decrementa Sy por 1;
22             iRDS ← calcule o índice linear (Sy, Sx) usando aux;
23         end
24     end
25     Retorne inDS, iRDS;
```

O Algoritmo 6 é uma descrição do pseudocódigo de um procedimento para determinar a porta de entrada e o índice de um roteador de destino (inDS e iRDS, respectivamente) em uma arquitetura NoC, com base na porta de saída de um

roteador de origem (*outUS*) e nas posições (Sx, Sy) dentro de uma matriz que representa a rede. A matriz é definida pelo número de linhas (nR) e colunas (nC).

Esse procedimento foi criado para a implementação de estratégias de roteamento eficientes em sistemas *many-core*, em que a eficácia da comunicação depende diretamente da capacidade de calcular rotas ótimas que minimizem o consumo energético e maximizem o desempenho. O cálculo do índice linear (iRDS) para determinar a localização exata do roteador de destino na matriz NoC é uma operação fundamental que permite a aplicação direta da métrica de custo de energia discutida anteriormente.

Ligando tal procedimento à equação de custo simplificado de energia total ($C_{\text{total}} = \sum_{i=1}^{n}(q_i \cdot C_{\text{inicial}})$), cada decisão de roteamento (representada pela escolha da porta de saída outUS e, consequentemente, pela porta de entrada *inDS* do próximo roteador) influencia o número de *links* (q_i) que um pacote deve atravessar. O custo energético inicial por link (C_{inicial}) multiplicado pelo número de *links* atravessados determina o custo energético associado à transmissão de pacotes através da rede.

Portanto, o pseudocódigo não apenas facilita o roteamento eficiente dentro da rede ao calcular as coordenadas de destino dos roteadores, mas também serve como base para a aplicação prática da equação de custo. Isso destaca a importância de estratégias de roteamento inteligentes na otimização do consumo de energia em arquiteturas NoC.

O Algoritmo 7 descreve o pseudocódigo de um método para calcular o número de *links* (nLinks) que um pacote atravessa dentro de uma arquitetura NoC, baseando-se nas posições de origem e destino do pacote dentro da rede. Tal cálculo é essencial para a avaliação do custo energético associado ao roteamento de pacotes, conforme apresentado pela equação de custo total simplificado de energia.

O procedimento identifica se o pacote permanece na mesma linha ou coluna dentro da matriz de roteadores (R). Caso permaneça, o número de *links* atravessados é zero ou calculado pela diferença absoluta entre as posições de origem e destino. Se o pacote se move tanto na horizontal quanto na vertical, o total de *links* é a soma das diferenças absolutas em cada direção.

Se um caminho está congestionado, a escolha do roteador depende das políticas de roteamento implementadas. Caso o usuário queira contabilizar os tempos de

Algoritmo 7: Pseudocódigo para Calcular o Número de Links

Entrada: Índice do *Grid* iG, Índice do Nó do Pacote iNP, Estrutura de Roteadores R
Saída : Número de Links $nLinks$

```
1  Função CalcNumLink(iG, iNP, R):
2      if R(iG).table(iNP,1) = R(iG).table(iNP,3) e
         R(iG).table(iNP,2) = R(iG).table(iNP,4) then
3          nLinks ← 0;
4      else
5          if R(iG).table(iNP,1) = R(iG).table(iNP,3) then
6              nLinks ← abs(R(iG).table(iNP,4) − R(iG).table(iNP,2));
7          else
8              if R(iG).table(iNP,2) = R(iG).table(iNP,4) then
9                  nLinks ← abs(R(iG).table(iNP,3) − R(iG).table(iNP,1));
10             else
11                 nLinks ← abs(R(iG).table(iNP,3) − R(iG).table(iNP,1))
                     + abs(R(iG).table(iNP,4) − R(iG).table(iNP,2));
12             end
13         end
14     end
15     Retorne nLinks;
```

espera ou os custos adicionais de tomar um caminho alternativo, é possível adicionar uma nova regra para essa métrica. Dessa forma, o simulador pode considerar o tempo de espera nos *buffers* e os custos adicionais de transmissão, avaliando o desempenho da NoC sob diferentes condições de tráfego e estratégias de roteamento. Isso permite uma análise detalhada das decisões de roteamento e seu impacto no desempenho geral do sistema.

4.3.6 Módulo de Geração de Resultados

O módulo de geração de resultados, um componente essencial do simulador NoC, é evidenciado na Figura 4.16. Ele converte dados da simulação em dois formatos principais: o "Modelo de Mapeamento"(Figura 4.7b) e os "Resultados Numéricos das Métricas"(Figura 4.7c). Para simplificar e facilitar a compreensão, por padrão, todos os algoritmos de mapeamento empregados na validação, ou

qualquer algoritmo de roteamento, produzem o "Mapa de Alocação de Tarefas", que é gerado através de algoritmos específicos que implementam diferentes estratégias de mapeamento. Essas estratégias levam em consideração as dependências vistas em um *DAG*, garantindo que o mapeamento das tarefas siga a estrutura do *DAG* desde a primeira tarefa até a última, respeitando os pesos e as dependências definidas. A geração dos mapas é feita de acordo com a estrutura do *DAG*, assegurando que as interdependências e a sequência das tarefas sejam mantidas, proporcionando uma alocação otimizada e coerente para a execução das aplicações na NoC.

Inicialmente, este é apresentado na forma de um vetor, em que os índices representam os PEs, distribuídos numericamente em ordem crescente conforme o tamanho do *grid* (Figura 4.7a). Após ser processado pelo módulo de geração de resultados, o mapa é então convertido em um formato visualmente claro na topologia *mesh*, acompanhado de tabelas com os resultados numéricos. O usuário pode escolher entre o melhor mapa ou todos os mapas gerados, variando de acordo com o número de iterações ou evoluções, dependendo da natureza dos algoritmos. O "melhor mapa"pode ser identificado com base nos critérios de parada escolhidos pelo usuário, pelas características dos algoritmos em uso ou pela métrica adotada. No caso do consumo de energia, o melhor mapa seria aquele que apresenta o menor valor gerado pela solução.

Este módulo possibilita ao usuário, a partir de resultados numéricos, convertê-los em formatos de saída, incluindo Excel e LaTeX, e, em seguida, gerar gráficos de comparação, como, por exemplo, estimativas do consumo de energia. Os usuários têm a liberdade de personalizar a visualização dos dados, configurando a exibição de resultados para atender às exigências específicas dos seus estudos.

Adicionalmente, o módulo permite criar representações gráficas que evidenciam a distribuição das tarefas na topologia *mesh*, funcionalidade ilustrada na Figura 4.7a. O propósito fundamental é gerar um mapa ótimo de alocação de tarefas, otimizado com base em métricas de interesse.

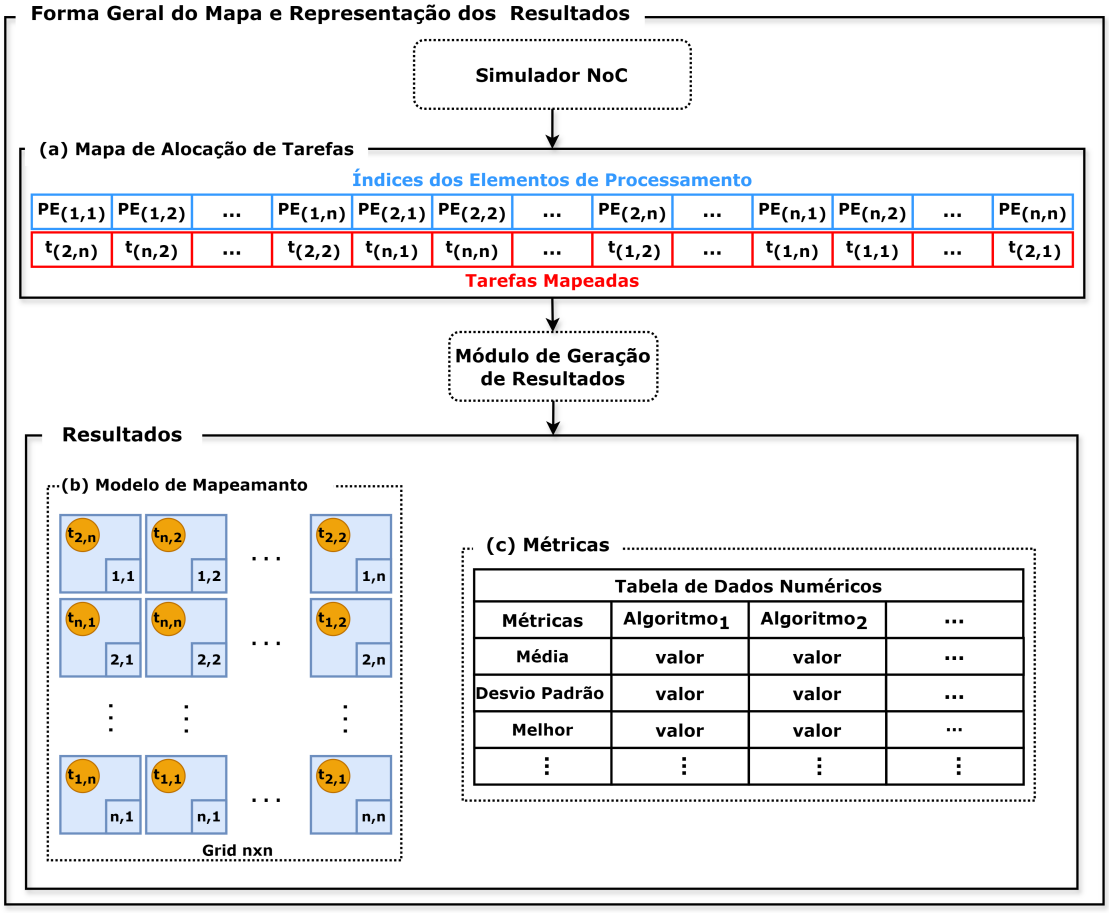

Figura 4.7 – Resultados Gerados pelo Simulador NoC.

4.4 Exemplo do Ambiente de Simulação

Após a descrição detalhada de cada componente da estrutura que constitui o ambiente de simulação, um exemplo de aplicação é ilustrado na Figura 4.8. A figura demonstra a sequência lógica de etapas desempenhadas por cada função do simulador, desde a inserção da aplicação até a geração e entrega dos pacotes.

O exemplo é categorizado em partes principais que descrevem o fluxo lógico da aplicação: na Figura 4.8A é detalhado o conjunto de configurações e definições, que inclui: o grafo de aplicações (Figura 4.8A_1), a matriz do grafo (Figura 4.8A_2), o mapa de alocação de tarefas (Figura 4.8 A_3), a estrutura da tabela de roteadores (Figura 4.8 A_4), e a abstração do conjunto de pacotes gerados (Figura 4.8 A_5). As Figuras 4.8B, C e D ilustram as iterações sucessivas do simulador, começando com

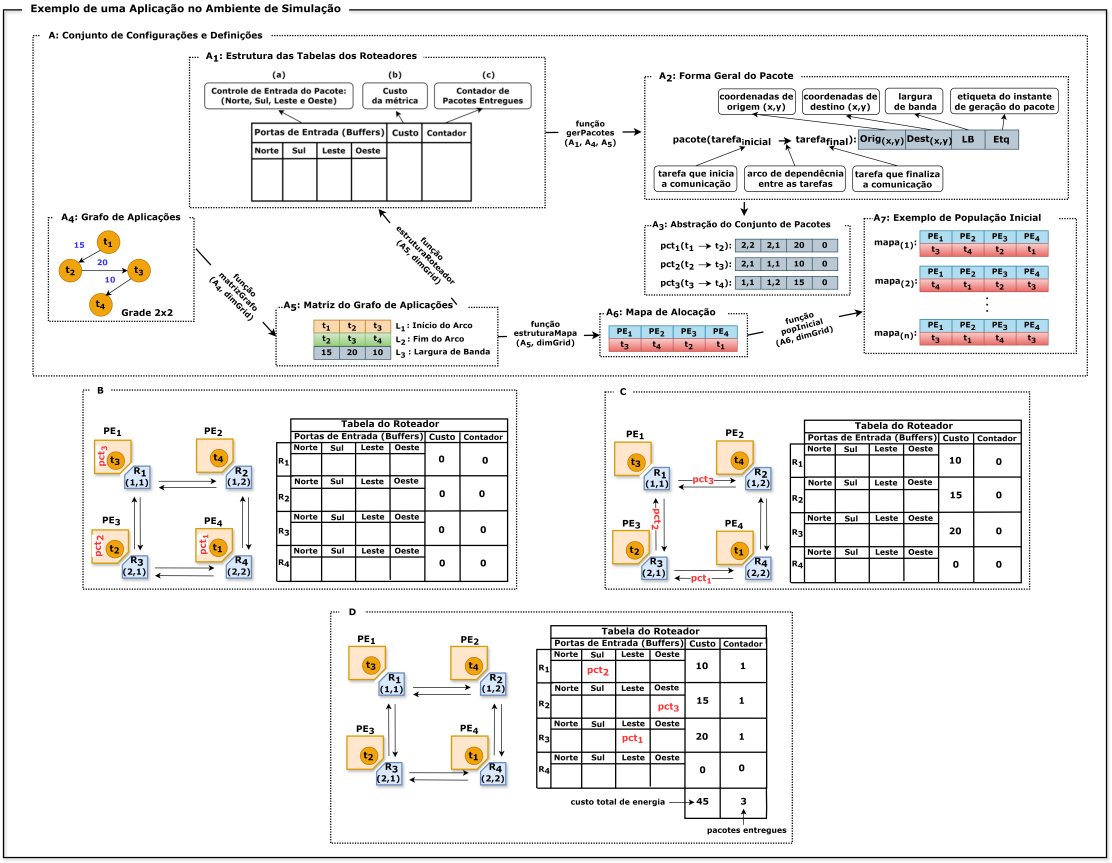

Figura 4.8 – Representação da Abstração da Arquitetura do Simulador.

a iteração zero (Figura 4.8B), quando o simulador é inicializado; prosseguindo para a primeira iteração (Figura 4.8C), em que os pacotes começam a ser processados; e concluindo com a iteração dois (Figura 4.8D), que exibe os pacotes em suas posições finais.

Cada uma dessas configurações e definições é gerada por funções de programação, que, na estrutura geral do simulador, são organizadas em módulos. A função `matrizGrafo(A1, dimGrid)` é utilizada no contexto da Subseção 4.3.2, tendo como parâmetros de entrada o grafo de aplicações e a dimensão do *grid*. Na sequência, a função `estruturaRoteador(A1, dimGrid)` recebe como parâmetros a matriz do grafo e a dimensão do *grid*, no contexto da Subseção 4.3.4. Da mesma forma, a função estrutura do mapa recebe como parâmetros a matriz do grafo de aplicações e a dimensão do *grid*.

A simulação inicia com a "Iteração 0"(Figura 4.8B), estabelecendo o ponto inicial para as operações subsequentes. Nesse estágio, são definidos a configuração da rede e os parâmetros de simulação, incluindo a inicialização dos *buffers* dos roteadores com os pacotes gerados conforme o grafo de aplicações. Tal procedimento inicial é fundamental, pois determina as condições sob as quais o sistema de NoC será simulado, influenciando as etapas seguintes e o desempenho global. A correta alocação dos pacotes nos *buffers* prepara a infraestrutura para o processo de roteamento, estabelecendo a base para a análise das iterações futuras.

Progredindo para a "Iteração 1"(Figura 4.8C), a simulação prossegue com a dinâmica de roteamento dos pacotes para seus destinos. Essa fase é instrumental para avaliar a eficácia dos algoritmos de roteamento aplicados, bem como para começar a compreender o impacto das decisões de mapeamento no desempenho da rede. Durante tal iteração, os custos associados à transferência de dados entre as tarefas são calculados e registrados, fornecendo uma medida inicial de eficiência e identificando possíveis gargalos na comunicação.

Finalmente, a "Iteração 2"(Figura 4.8D) revela os resultados da simulação, com todos os pacotes tendo alcançado seus destinos finais. Essa etapa final é fundamental para a avaliação geral do desempenho do sistema, incluindo o consumo de energia e a eficiência do mapeamento de tarefas. A análise desses resultados permite uma reflexão crítica sobre as estratégias de mapeamento utilizadas, abrindo caminho para otimizações futuras que possam melhorar a taxa de entrega de pacotes e reduzir o consumo energético.

A sequência de iterações descrita detalha de forma simplificada o fluxo de operações dentro do ambiente simulado de NoC, desde a preparação inicial até a conclusão da transmissão de dados. Tal abordagem estruturada não só facilita a compreensão do comportamento da rede sob diferentes configurações, mas também destaca a importância de um mapeamento de tarefas e roteamento eficiente para o desempenho geral do sistema. Através dessa análise, fica evidente que o equilíbrio entre a eficácia do roteamento e o consumo total de energia na rede é essencial para o desenvolvimento de abordagens que sejam tanto confiáveis quanto eficientes.

4.5 Considerações finais

Neste capítulo, detalhou-se a metodologia aplicada ao desenvolvimento e operação de um simulador de alto nível para NoCs, proposto neste projeto de doutorado. A estruturação em módulos mostrou-se essencial para lidar com a complexidade do problema, permitindo organizar os processos de simulação em componentes gerenciáveis. Cada módulo, desde a configuração inicial de parâmetros até a geração de resultados, constitui-se em um conjunto de etapas essenciais para simular o mapeamento e execução de aplicações em NoCs, conduzidas pelo bloco principal, o *Simulador NoC*.

Os mecanismos de mapeamento podem ser implementados através de linguagens e *frameworks* adequados para o desenvolvimento de sistemas *many-core* paralelos. O simulador identifica a combinação mais apropriada de opções de mapeamento e roteamento. Essas opções devem ser especificadas na linguagem de programação paralela adotada, e a interface com o sistema operacional é realizada através de mecanismos de tempo de execução, otimizando o desempenho da aplicação.

Contudo, a integração entre o ambiente de simulação e a variedade considerável de algoritmos adotados para experimentação introduziu desafios, muitas vezes resolvidos com intervenções manuais em determinados pontos do processo de simulação. Essa complexidade decorre, em parte, da necessidade de interação com sistemas reais e da adaptação de ferramentas existentes, que, por vezes, exigem ajustes e configurações além das capacidades de automatização atualmente disponíveis no simulador. Essas intervenções manuais são indispensáveis para garantir que os dados gerados reflitam com precisão o comportamento das NoCs sob condições diversas. Isso ocorre porque o simulador, embora eficiente, ainda não está totalmente otimizado para todas as situações experimentais, exigindo, portanto, ajustes manuais para a validação dos resultados. Assim, apesar desses desafios, pode-se considerar que a metodologia de simulação proposta atingiu os objetivos de oferecer um ambiente simplificado e eficiente para a simulação de sistemas *many-core* baseados em NoCs, servindo como base conceitual para a exploração de estratégias de otimização e para o avanço do conhecimento na área.

Capítulo 5

Resultados e Discussões

Neste capítulo, apresenta-se uma série de experimentos, inicialmente visando validar a confiabilidade dos dados produzidos pelo Simulador NoC, seguido por experimentos do tipo DSE (Exploração do Espaço de Opções de Projeto), procedimento que evidencia as funcionalidades e potencialidades da ferramenta proposta nesta tese.

Os resultados referentes à condução de experimentos, utilizando a metodologia desenvolvida como parte deste projeto de doutorado (Capítulo 4), são apresentados na sequência. As seções seguintes estão organizadas conforme os seguintes conteúdos:

a) Inicialmente, na Seção 5.1, são detalhados a **metodologia e os parâmetros** empregados na condução dos experimentos, estabelecendo as bases para as avaliações realizadas;

b) O primeiro experimento, descrito na Seção 5.2, teve como objetivo validar a eficiência do simulador quanto à capacidade de gerar resultados consistentes com os do simulador NoCTweak, que utiliza uma métrica mais realista (CMOS). Para isso, foram comparadas as curvas das estimativas de consumo de energia dos algoritmos rodados em ambos os simuladores, verificando a similaridade das tendências observadas;

c) O segundo experimento, descrito na Seção 5.3, trata da exploração do espaço de projeto (DSE) em relação às estratégias de mapeamento de tarefas, gerando resultados a partir da variação nos algoritmos de mapeamento de objetivo único e recombinação de seus parâmetros, mantendo fixa a estratégia de roteamento de pacotes com a utilização do algoritmo XY. Buscou-se identificar as melhores opções de mapeamento que otimizem a estimativa de consumo de energia;

d) No terceiro experimento, descrito na Seção 5.4, adotou-se uma abordagem de DSE alternativa, variando os algoritmos de roteamento e fixando o uso de uma versão adaptada para mapeamento de tarefas do algoritmo GA, permitindo uma análise comparativa do impacto das diferentes estratégias de roteamento na métrica de consumo de energia estabelecida.

Apesar de a análise, compilação e discussão dos dados obtidos serem essenciais para evidenciar as complexidades e implicações das diversas técnicas de mapeamento e roteamento, é importante ressaltar que o objetivo principal dos experimentos apresentados reside na validação da eficácia do simulador para os objetivos estabelecidos pelo projeto, especialmente no que diz respeito a viabilizar estudos de DSE, envolvendo o contexto e parâmetros de configuração adotados.

5.1 Experimentos: Configuração e Parâmetros

Nesta seção, são apresentados os dados relevantes para a condução dos experimentos descritos neste capítulo, possibilitando a eventual reprodução dos mesmos.

5.1.1 Grafos Representando Aplicações

Nesta ferramenta, eventuais aplicações que rodariam em sistemas *many-core* são representadas por DAGs, que são estruturas compostas por tarefas interconectadas. As relações entre essas tarefas são modeladas por arcos, que possuem pesos específicos atribuídos a cada conexão. Esses pesos representam a quantidade de dados, em uma forma abstrata, a ser transmitida pelo referido *link*. Esses grafos constituem a base para a geração dos pacotes que circulam pela rede, fundamentais na simulação do comportamento e da comunicação em NoCs. Os pacotes de

dados, essenciais para a simulação, derivam das conexões estabelecidas entre as tarefas. Na Tabela 4, são descritas as características gerais dos grafos utilizados nos experimentos e o número de pacotes gerados a partir de cada um deles.

Tabela 4 – Características dos Grafos de Tarefas Utilizados nos Experimentos

Nome	**Grid**	**Nº de Tarefas**	**Nº de Arcos**	**Pesos**	**Nº Pacotes**
Grafo$_{9}$	3x3	9	9	0-49	9
Grafo$_{16}$	4x4	16	17	0-49	17
Grafo$_{25}$	5x5	25	26	0-49	26
Grafo$_{36}$	6x6	36	42	0-49	42
Grafo$_{49}$	7x7	49	60	0-49	60
Grafo$_{64}$	8x8	64	76	0-49	76
Grafo$_{81}$	9x9	81	98	0-49	98
Grafo$_{100}$	10x10	100	120	0-49	120

A identificação dos grafos é determinada pelo número total de tarefas que os compõem, correspondendo ao número de nós no grafo. A configuração da rede é descrita por um *grid* (nxn), representando o número de linhas e colunas, e, por consequência, o número de nós em cada dimensão. Assim, o total de tarefas, equivalente ao número de nós, define a capacidade máxima do grafo. Importante destacar que, neste contexto, não é utilizada a clusterização de tarefas, mas sim a alocação unitária, onde cada tarefa é alocada individualmente. Cada tarefa representa um processo ou atividade na simulação, e seu aumento implica maior complexidade de processamento e comunicação na NoC.

Os arcos do grafo indicam as conexões diretas entre tarefas, essenciais para modelar a comunicação entre pacotes na rede. Tais conexões, que representam as dependências entre as tarefas no grafo, determinam a quantidade de pacotes gerados para transmissão, afetando o volume de comunicação simulado. Os pesos atribuídos aos arcos definem a quantidade de dados abstratos a serem transmitidos, variando de 0 a 49 (conforme o padrão do *software TGFF*), e influenciam diretamente na geração de pacotes na simulação. Cada pacote representa uma unidade de dados transmitida de uma tarefa para outra, simulando o fluxo de informações na NoC.

5.1.2 Topologia da NoC

A topologia *mesh* foi selecionada para os experimentos devido à sua abordagem conceitual simplificada e direta, que facilita o entendimento da rede como um todo. Caracterizada por uma estrutura em *grid*, na qual cada nó computacional está diretamente conectado aos seus vizinhos adjacentes, exceto nas bordas, a topologia *mesh* suporta comunicações diretas e oferece múltiplas rotas entre nós, contribuindo para a redução da latência e do risco de congestionamentos. Sua escalabilidade, especialmente relevante em grafos de dimensões maiores do que os comumente descritos na literatura, também foi um fator decisivo na escolha dessa topologia. Além disso, essa topologia permite a avaliação da eficiência da rede sob diferentes estratégias de roteamento de pacotes e mapeamento de tarefas, impactando significativamente no desempenho da rede.

5.1.3 Algoritmos de Mapeamento

Conforme descrito na Seção 4.2, o simulador foi desenvolvido para suportar uma diversidade de algoritmos de mapeamento de aplicações, facilitando a avaliação e o entendimento conceitual dentro de um ambiente controlado. No ambiente de simulação proposto, esses algoritmos podem ser da própria base de dados já implementada, provenientes da base de algoritmos da plataforma *PlatEMO*, ou ainda de implementações personalizadas pelos usuários.

Nos experimentos relatados neste capítulo, a escolha de algoritmos de mapeamento foi baseada em suas propriedades únicas, abrangendo desde heurísticas até técnicas baseadas em aleatoriedade. Tal seleção visa oferecer uma análise compreensiva das várias estratégias de mapeamento disponíveis, explorando um amplo espectro de possibilidades e destacando as potenciais vantagens e limitações de cada abordagem em termos de eficácia e eficiência na simulação. Nesse sentido, cinco algoritmos adaptados para o mapeamento de tarefas foram selecionados para a condução dos experimentos, cada um com especificidades e variações próprias de parâmetros, conforme descrito a seguir:

a) *Tabu Search (TS)*: Adotando uma implementação própria, o *Tabu Search* oferece uma abordagem heurística para solucionar problemas de otimização,

evitando ciclos na busca por soluções (OLIVEIRA; CARVALHO; KREUTZ, 2021).

b) *Simulated Annealing (SA)*: Também com uma implementação própria, este algoritmo visa explorar de maneira eficaz o espaço de soluções com sua estratégia de perturbação modificada. Essa abordagem consiste em três métodos principais (TAJARY; MORSHEDLOU, 2022):

   1. *Swapping Two Nodes*: Dois roteadores são selecionados e as tarefas que eles contêm são trocadas entre si;
   2. *Circulating Three Nodes*: Três roteadores são utilizados em um ciclo de permuta de tarefas;
   3. *Swapping Four Nodes, Two by Two*: Quatro roteadores são selecionados para uma dupla troca de tarefas.

c) *Genetic Algorithm (GA)*: Um dos algoritmos mais recorrentes na literatura para otimização, ele é utilizado a partir da base de dados da plataforma PlatEMO, aproveitando sua capacidade de explorar eficientemente o espaço de soluções (MESSAOUDI et al., 2020).

d) *Ant Colony Optimization (ACO)*: Inspirado no comportamento das formigas na natureza, este algoritmo é empregado para otimizar rotas de mapeamento e está disponível através da plataforma PlatEMO (SHARMA et al., 2023b).

e) *Particle Swarm Optimization (PSO)*: Baseado no comportamento de enxames, o ele é utilizado para encontrar soluções ótimas de mapeamento, aproveitando as funcionalidades da plataforma PlatEMO (SALEEM et al., 2023).

Para cada algoritmo estudado, foram investigadas 27 diferentes configurações paramétricas. Essas configurações resultaram do cruzamento de parâmetros específicos para cada técnica, como a temperatura inicial no *SA*, o tamanho da lista tabu no *TS* e os tipos de mutação no *GA*, cada um combinado com outros parâmetros para alcançar o total de combinações.

Na Tabela 5, são descritos os parâmetros adaptados para o algoritmo *TS*, focando em três componentes: a *Tabu List*, o *Number of Iterations Without Im-*

provement e o *Number of Neighbors Generated*. A lista tabu controla a memória do algoritmo, proibindo certas soluções recentes para evitar ciclos de busca.

O critério de iterações sem melhoria determina quando o algoritmo deve mudar sua direção de busca após não obter melhorias significativas, incentivando a exploração de novas áreas do espaço de solução. Por fim, o número de vizinhos gerados ajusta a abrangência da busca local, com um maior número de vizinhos permitindo uma exploração mais detalhada nas proximidades da solução atual. Esses parâmetros são fundamentais para balancear a exploração e a busca detalhada dentro do espaço de soluções.

Tabela 5 – Parâmetros Variados no Algoritmo - TS

Lista Tabu	Iterações Sem Melhoria	Vizinhos Gerados
Nº de Tarefas	10	5
Nº de Tarefas*2	20	10
Nº de Tarefas*3	50	15

A Tabela 6 detalha os parâmetros ajustados na variante do algoritmo *SA*: a *Initial Temperature*, a *Minimum Temperature* e o *Temperature Reduction Factor*. A *Initial Temperature* define a abrangência inicial da exploração do espaço de soluções, enquanto a *Minimum Temperature* determina o ponto de finalização do processo de busca, garantindo que a otimização continue até atingir um resultado satisfatório. O *Temperature Reduction Factor* modula a velocidade de diminuição da temperatura durante a execução do algoritmo, facilitando a transição de uma ampla exploração inicial para uma focalizada convergência nas fases finais. Esses parâmetros promovem um equilíbrio entre a descoberta de novas áreas do espaço de soluções e a otimização das melhores soluções encontradas. Esse algoritmo, cujas particularidades em relação ao método tradicional foram discutidas anteriormente neste capítulo, destaca-se por sua abordagem modificada na função de perturbação.

Tabela 6 – Parâmetros Variados no Algoritmo - SA (Tajary et al. (2022)).

Temperatura Inicial	Temperatura Mínima	Redução da Temperatura
100	0.01	0.85
150	0.05	0.92
200	0.1	0.95

No contexto de algoritmos genéticos, a estratégia evolutiva adotada enfatizou o uso de mutações, fundamentais para introduzir diversidade na população de soluções. Essa abordagem permite ao algoritmo explorar novas áreas do espaço de busca e evitar a convergência prematura para máximos locais. A Tabela 7 ilustra os parâmetros para três tipos de operações de mutação, com nove configurações para cada tipo, definidas pelas taxas de mutação e pela distância de mutação.

Tabela 7 – Parâmetros Variados no Algoritmo - GA

Tipo de Mutação	**Taxa de Mutação (*proM*)**	**Distância de Mutação (*distM*)**
Mutação por Troca	0.02	10, 20, 50
Mutação por Inserção	0.05	10, 20, 50
Mutação por Inversão	0.01	10, 20, 50

A estratégia evolutiva aplicada nos experimentos com o *GA* começou com a *Swap Mutation*, um método particularmente eficaz em problemas de permutação. Nesse processo, duas posições, representando unidades de processamento (*PEs*) no mapa de alocação de tarefas, são selecionadas aleatoriamente, e as tarefas atribuídas a essas posições são trocadas. Esse tipo de mutação ajuda a explorar novas configurações de alocação de tarefas, potencialmente buscando melhorar o consumo de energia (ALVES; OLIVEIRA; NETO, 2015).

Em seguida, a *Insertion Mutation* foi aplicada. Nessa abordagem, um gene, ou tarefa, é aleatoriamente escolhido e reinserido em uma nova posição, causando um deslocamento nos demais genes. Essa técnica é particularmente útil em cenários em que a ordem das tarefas afeta o desempenho geral, permitindo uma reorganização eficiente das tarefas para otimizar a execução (BHATT; CHAUHAN, 2015).

Por último, a *Inversion Mutation* foi utilizada, na qual uma subseção específica do cromossomo é selecionada e a ordem dos genes, ou tarefas, dentro dela é invertida. Tal método pode ser vantajoso para manter agrupamentos de tarefas que colaboram bem juntas, potencializando a eficiência da execução ao preservar sequências produtivas de alocação enquanto explora novas combinações globais (KATOCH; CHAUHAN; KUMAR, 2021).

Na Tabela 8 para o *ACO*, são detalhados os parâmetros das configurações: a *Pheromone Evaporation Rate*, a *Amount of Pheromone* e a *Initial Influence of the Pheromone*. A taxa ajusta a persistência das trilhas de feromônio, com valores altos promovendo a exploração por reduzir a influência de caminhos antigos. A

quantidade define o reforço das trilhas bem-sucedidas, orientando a busca futura, enquanto a influência estabelece o nível de feromônio no início da busca, influenciando a direção inicial da exploração. Esses parâmetros são essenciais para balancear entre a exploração de novas soluções e aprimorar os conhecimentos existentes, visando à otimização efetiva do algoritmo.

Tabela 8 – Parâmetros Variados no Algoritmo - ACO

Taxa de Evaporação	Quantidade de Feromônio	Influência Inicial
0.1	1	1
0.5	2	5
0.9	3	10

Na Tabela 9, são definidas as configurações para o PSO, focando no *Inertia Weight (W)*, *Cognitive Coefficient* (C_1) e *Social Coefficient* (C_2). Esses parâmetros ajustam o equilíbrio do algoritmo entre explorar novas áreas do espaço de soluções e aprimorar as melhores soluções já encontradas. O W regula a velocidade das partículas, influenciando diretamente quão longe elas podem explorar fora de suas posições atuais. O C_1 afeta a tendência das partículas de seguir suas próprias experiências passadas para aprimorar suas soluções, enquanto o C_2 determina a inclinação das partículas em direção à melhor solução encontrada pelo enxame, favorecendo a exploração coletiva de soluções promissoras. A interdependência entre esses parâmetros é fundamental para a performance do PSO, permitindo que o algoritmo equilibre efetivamente a inovação com a otimização das soluções existentes.

Tabela 9 – Parâmetros Variados no Algoritmo - PSO

Peso de Inércia (W)	Coeficiente Cognitivo (C_1)	Coeficiente Social (C_2)
0.4	1.5, 2.0, 2.5	1.5, 2.0, 2.5
0.6	1.5, 2.0, 2.5	1.5, 2.0, 2.5
0.8	1.5, 2.0, 2.5	1.5, 2.0, 2.5

5.1.4 Algoritmos de Roteamento

Os algoritmos de roteamento desempenham uma função essencial na determinação da eficácia e eficiência das comunicações em NoCs (FENG, 2023). No

contexto dos experimentos descritos neste capítulo, após a seleção do Algoritmo Genético (GA) como base para as estratégias de mapeamento, **seis algoritmos de roteamento** foram selecionados para condução dos experimentos. Os algoritmos selecionados para os experimentos, e detalhados na Seção 2.3.2, são os seguintes:

a) *West-First (WF)*: Esse algoritmo proíbe pacotes de se moverem para o leste após terem se movido para o oeste, reduzindo a possibilidade de *deadlocks* e minimizando o caminho percorrido (JEANG et al., 2008);

b) *Negative-First (NF)*: Similar ao WF, esse algoritmo restringe o movimento dos pacotes na direção negativa após terem adotado uma direção positiva, visando otimizar a rota e evitar bloqueios (SOMISETTY et al., 2022);

c) *North-Last (NL)*: Nesse algoritmo, os pacotes podem se mover em qualquer direção, exceto para o norte, depois de terem iniciado movimento para o sul, leste ou oeste, priorizando rotas que minimizem o uso do tráfego norte (LIU, 2021);

d) *Odd-Even (OE)*: Esse é um algoritmo adaptativo que determina a rota baseada em regras específicas, relacionadas às paridades dos destinos e posições atuais, efetivamente reduzindo conflitos e congestionamentos (HENTGES; ABDELREHIM, 2021; ZHANG et al., 2009a);

e) *XY (Determinístico)*: Um algoritmo simples e determinístico que primeiro roteia pacotes ao longo do eixo X até alcançar a coluna do destino e, em seguida, move-se ao longo do eixo Y até o destino. É valorizado por sua previsibilidade e simplicidade (AN et al., 2021);

f) *XYX (Experimental)*: Uma variação do clássico XY, que introduz uma etapa adicional, permitindo movimentos adicionais ao longo do eixo X, após alcançar a coluna correta, para explorar caminhos alternativos em cenários específicos (SONG; LIN, 2019).

A escolha desses algoritmos para os experimentos foi motivada pela necessidade de compreender o comportamento da rede sob diferentes estratégias de roteamento, desde abordagens determinísticas simples até métodos adaptativos mais complexos. Essa análise visa identificar as técnicas que melhor equilibram o desempenho,

a eficiência energética e a confiabilidade em diversas configurações de carga e topologia de rede.

Conforme mencionado anteriormente, para avaliar as estratégias de roteamento selecionadas, será fixado o GA) como estratégia de mapeamento de tarefas na NoC. Nesse caso, os dados da Tabela 10 apresentam os parâmetros que serão combinados nesses experimentos.

Tabela 10 – Parâmetros de GA para experimentos variando estratégia de roteamento

Tipo de Mutação	**Taxa de Mutação (*proM*)**
Mutação por Troca	0.1, 0.02, 0.05
Mutação por Inserção	0.1, 0.02, 0.05
Mutação por Inversão	0.1, 0.02, 0.05

5.1.5 Métrica de Resultados Considerada

Nos experimentos realizados, adotou-se a métrica da estimativa do consumo de energia, obtida pelo método simplificado descrito na Subseção 4.3.5. Os dados foram obtidos para os algoritmos sob condições padronizadas, considerando 100 soluções por execução e 50.000 evoluções, quando aplicável. Após 50 execuções independentes, calcula-se a média do consumo de energia (Média), selecionando-se os melhores resultados de cada conjunto de 100 soluções. Essa metodologia permite identificar tanto a eficiência energética média quanto a variabilidade entre execuções, empregando-se o desvio padrão para quantificação dessa variabilidade. O destaque incide sobre o valor mínimo de consumo de energia encontrado (Melhor), que evidencia a solução mais eficiente dentre todas as avaliadas. Os resultados obtidos são apresentados numericamente com precisão de duas casas decimais.

As conclusões derivadas dos resultados numéricos das métricas sugerem, para uma dada configuração de NoC e aplicação específica em análise, a combinação mais eficaz de algoritmos de mapeamento e roteamento, bem como ajustes de parâmetros aplicáveis. Lembrando que, para essa ferramenta, uma aplicação é caracterizada pelo tipo de grafo utilizado e o número de tarefas, além dos pesos

atribuídos às comunicações entre essas tarefas. Identificar a combinação ideal envolve analisar o desempenho dos algoritmos em termos de critérios como eficiência energética (potência) e eficácia (Melhor resultado), com base nos dados fornecidos. A meta é otimizar o processo de mapeamento e roteamento de tarefas dentro da rede representada pelo grafo, considerando a complexidade das interações entre as tarefas e as demandas de comunicação, para melhorar a performance geral da aplicação a ser executada em um sistema *many-core* baseado em NoC.

5.1.6 Plataforma de Condução dos Experimentos

Os experimentos foram realizados em duas configurações distintas de computadores. O primeiro, com um processador *AMD Ryzen 7 5700G* com *Radeon Graphics* de 3,80 GHz e 16 gigabytes de RAM, rodando o sistema operacional *Windows 10 Pro*, versão 21H2. O segundo computador estava equipado com um processador *Intel Core i7* de 3,60 GHz, 32 gigabytes de RAM e uma placa gráfica *NVIDIA GeForce RTX 3080*, operando sob o sistema *Linux Ubuntu 18.04*. Em ambos os casos, foram utilizados o software *MATLAB* em sua versão básica e o *Microsoft Excel* para análise de dados, além de *SystemC* e *C++* para os simuladores específicos.

5.2 Experimento para Validação do Simulador e do Método

Conforme mencionado anteriormente, esse conjunto de experimentos buscou validar a confiabilidade do simulador e do método desenvolvido para o *Simulador NoC* no que diz respeito às estimativas de consumo de energia. Para isso, foram comparados os resultados gerados com aqueles produzidos pelo simulador *NoCTweak* (TRAN; BAAS, 2012).

Em princípio, esse experimento não visa à comparação de valores absolutos produzidos pelas duas ferramentas, mas sim à observação de curvas que reflitam uma **comparação qualitativa entre duas ou mais opções de projeto**.

Os resultados, apresentados na Tabela 11 e ilustrados nos gráficos das Figuras 5.1 a 5.8, referem-se a experimentos com grafos representando aplicações de

complexidade variada ($Grafo_9$ até $Grafo_{100}$). Esses experimentos foram realizados seguindo uma metodologia rigorosa para garantir a confiabilidade e precisão dos dados, gerando resultados numéricos para a métrica de estimativa de consumo de energia simplificada.

Os valores para a métrica foram calculados utilizando-se a ferramenta proposta, adotando o algoritmo de roteamento XY e os algoritmos de mapeamento previamente mencionados, assim como os algoritmos encontrados nas implementações originais dos simuladores NoCTweak (NMAP e RANDOM) e NoCMap (BB e SA tradicional). Adicionalmente, os melhores mapeamentos gerados no simulador deste projeto foram avaliados no NoCTweak, empregando-se, nesse caso, a biblioteca CMOS para o cálculo da potência. Ambos os simuladores utilizaram metodologias e parâmetros padrão para os algoritmos, conforme descrito nas publicações pertinentes. padrão para os algoritmos, conforme descrito nas publicações pertinentes.

Tabela 11 – Valores de Energia e Potência para Validação dos Resultados

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**	**NoCTweak**		**NoCmap**	
		GA	**ACO**	**PSO**	**Busca Tabu**	**SA (Tajary et al.,2022)**	**NMAP**	**RANDOM**	**BB**	**SA**
$Grafo_9$	**Energia**	350.00	450.00	350.00	350.00	350.00	458.00	411.00	430.00	467.00
	Potência	15.49	18.18	15.44	15.27	16.11	17.80	16.46	17.52	17.90
$Grafo_{16}$	**Energia**	486.00	858.00	568.00	516.00	473.00	1095.00	882.00	836.00	997.00
	Potência	24.67	37.63	26.27	26.64	25.90	35.98	32.50	31.46	34.28
$Grafo_{25}$	**Energia**	968.00	2052.00	1891.00	1121.00	905.00	2758.00	2036.00	2100.00	1908.00
	Potência	46.16	58.93	64.86	53.95	44.95	83.08	67.71	69.10	65.66
$Grafo_{36}$	**Energia**	1616.00	3985.00	3898.00	2012.00	1677.00	5059.00	3710.00	3993.00	3804.00
	Potência	70.55	116.32	114.85	77.40	71.46	137.46	110.86	116.80	111.99
$Grafo_{49}$	**Energia**	1987.00	5165.00	4787.00	2585.00	2214.00	5528.00	5123.00	5299.00	5041.00
	Potência	90.47	156.33	149.68	102.89	96.28	163.87	156.25	159.38	153.33
$Grafo_{64}$	**Energia**	3280.00	8855.00	8798.00	4368.00	3730.00	9106.00	9004.00	9020.00	8599.00
	Potência	128.16	233.77	234.44	148.99	136.54	240.57	237.06	238.67	229.15
$Grafo_{81}$	**Energia**	4318.00	11213.00	11398.00	6002.00	4887.00	12479.00	11282.00	11270.00	10899.00
	Potência	207.82	267.47	263.35	174.66	158.97	274.48	261.94	261.77	255.61
$Grafo_{100}$	**Energia**	6412.00	18793.00	19254.00	9000.00	7514.00	19745.00	19183.00	19759.00	19755.00
	Potência	146.17	207.23	214.68	237.73	151.22	220.77	211.90	212.06	212.06

Os mapas de alocação de tarefas otimizados no ambiente do simulador proposto foram usados como parâmetros de entrada no NoCTweak para avaliar a potência, permitindo, assim, uma análise comparativa da eficiência energética e do consumo de energia. Esse processo viabilizou a validação da confiabilidade dos resultados produzidos pelo simulador desenvolvido.

No gráfico da Figura 5.1, que trata de experimentos com uma aplicação de pequeno porte, podem ser observadas duas curvas: uma de energia (gerada pelo Simulador NoC) e outra de potência (gerada pelo NoCTweak). O objetivo dessa análise é identificar a existência de uma correlação entre elas. A correlação positiva

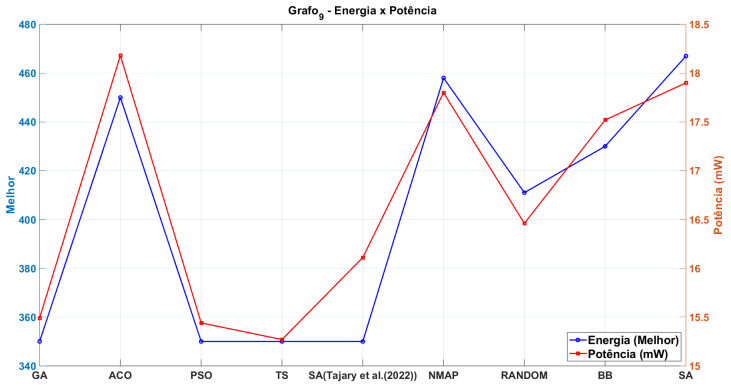

Figura 5.1 – Gráfico de Correlação - $Grafo_9$

(ou negativa) seria indicada por curvas que descem (ou sobem) juntas, refletindo influências positivas (ou negativas) da combinação de algoritmos e parâmetros adotados. Variações ou inversões de tendências nessas curvas poderiam sugerir diferenças significativas na maneira como cada algoritmo influencia a métrica de consumo de energia.

Para o caso específico ($Grafo_9$), a consistência no comportamento das curvas indica que os resultados produzidos por ambos os simuladores são coerentes, sugerindo a confiabilidade dos dados produzidos pelo Simulador NoC deste projeto.

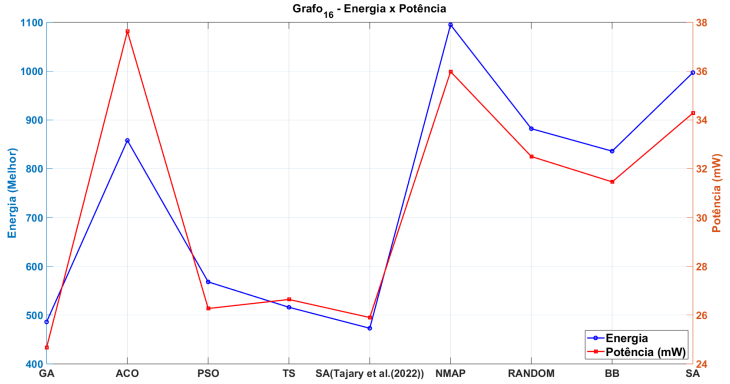

Figura 5.2 – Gráfico de Correlação - $Grafo_{16}$

Analisando o gráfico ilustrado na Figura 5.2, as curvas de energia e potência exibem comportamento semelhante, indicado pelo movimento conjunto em diversos pontos. Essa tendência sugere que os algoritmos produzem os mesmos efeitos

na estimativa de consumo de energia em ambos os simuladores, confirmando a validação dos experimentos.

Em análise similar, as curvas apresentadas nas Figuras 5.3, 5.4, 5.5, 5.6 e 5.7, referentes a aplicações de complexidade crescente, exibem as mesmas características observadas nos casos anteriores, reforçando a confiabilidade dos resultados produzidos pelo simulador proposto, quando comparado com um simulador de referência.

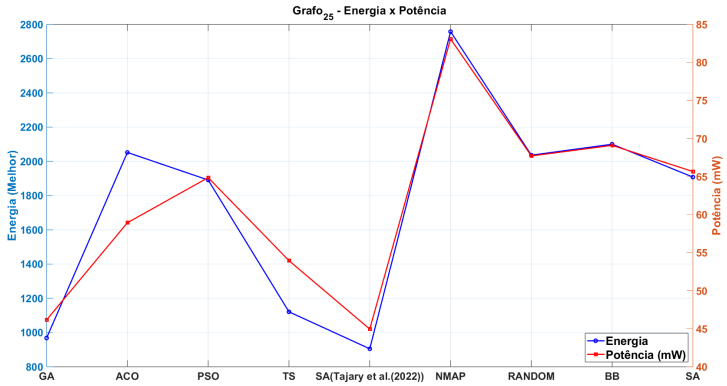

Figura 5.3 – Gráfico de Correlação - $Grafo_{25}$

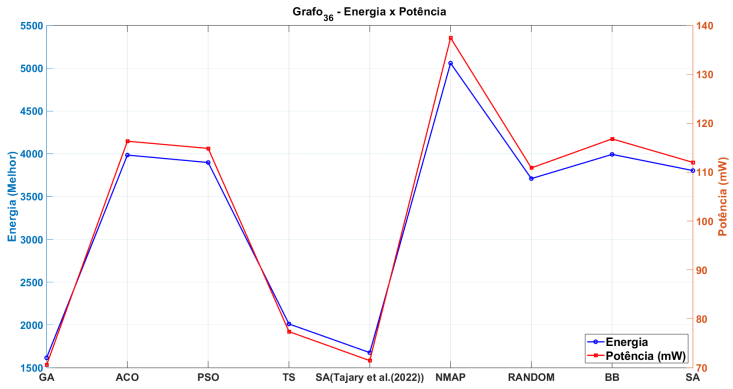

Figura 5.4 – Gráfico de Correlação - $Grafo_{36}$

Para complementar a análise, deve-se observar no gráfico da Figura 5.8 que as curvas apresentam comportamento similar aos casos anteriores. No entanto, há uma divergência clara na seção entre o terceiro e o quarto ponto do eixo horizontal, onde a Energia tem uma queda acentuada enquanto a Potência aumenta

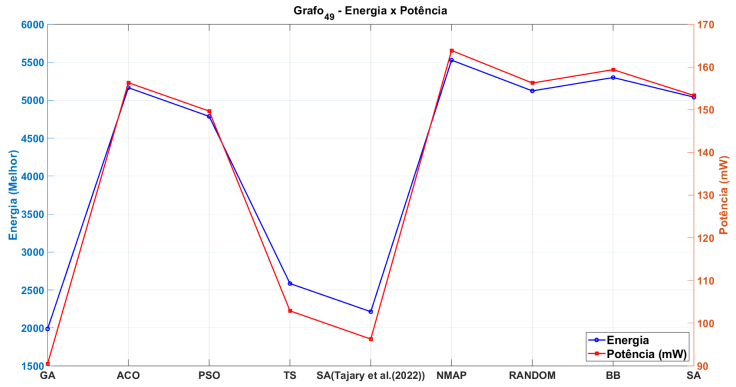

Figura 5.5 – Gráfico de Correlação - Grafo_{49}

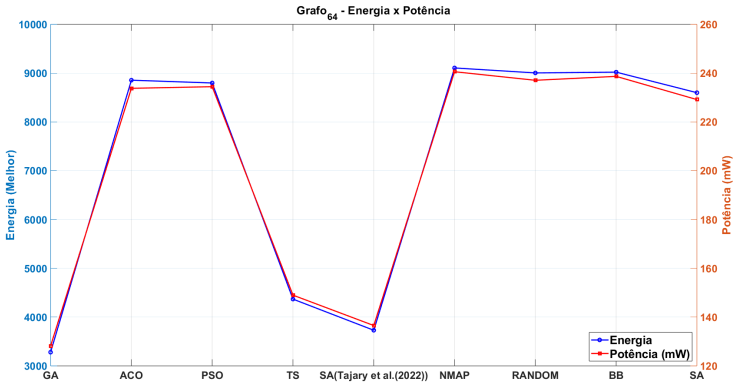

Figura 5.6 – Gráfico de Correlação - Grafo_{64}

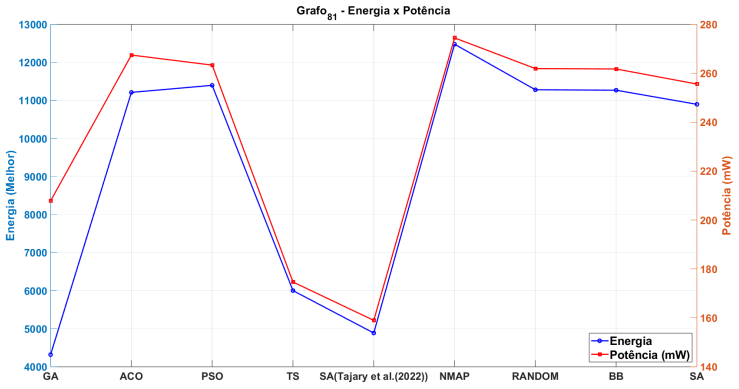

Figura 5.7 – Gráfico de Correlação - Grafo_{81}

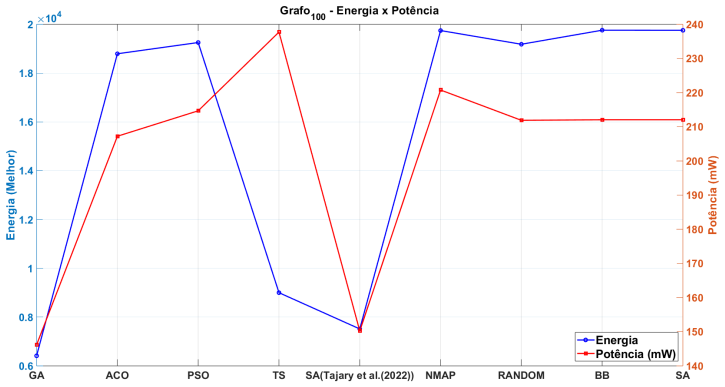

Figura 5.8 – Gráfico de Correlação - Grafo_{100}

ligeiramente. Essa discrepância pontual pode ter duas explicações teóricas: *i)* uma diferença significativa de comportamento entre os dois simuladores, devido às características da aplicação de grande porte (100 nós de tarefas); *ii)* uma possível falha no processamento ou compilação dos dados pelo Simulador NoC.

Considerando que esse é um único ponto de divergência entre muitos outros que exibiram o comportamento esperado, pode-se inferir que o simulador desenvolvido apresenta resultados confiáveis para os parâmetros considerados nesta pesquisa. No entanto, o motivo para a geração desse ponto divergente merece ser investigado mais profundamente, o que deverá ser abordado em possíveis versões futuras da ferramenta.

Em resumo, os gráficos mostram uma correlação entre consumo de energia e potência, apesar das variações no número de tarefas e das particularidades dos simuladores usados nas medições. As variações detectadas são esperadas dentro da complexidade das simulações e não comprometem a confiabilidade geral dos resultados. A metodologia de análise de correlação facilita a identificação de padrões essenciais para a seleção de algoritmos, visando à eficiência energética em contextos reais.

5.3 Experimento de DSE com Várias Estratégias para Mapeamento e Única para Roteamento

Os experimentos desta seção constituem um exemplo típico de exploração de opções de projeto (DSE), avaliando a eficácia relativa entre os algoritmos de mapeamento listados na Subseção 5.1.3, em combinação com uma estratégia fixa de roteamento, no caso o algoritmo XY. A avaliação envolveu experimentos com 27 variações de parâmetros para cada algoritmo de mapeamento, conforme descrito nas Tabelas 12 a 15. Dentre as tabelas de resultados geradas, quatro são discutidas nesta subseção, identificadas como "configuração n", ilustrando comportamentos e tendências identificadas a partir dos resultados obtidos. As demais 23 tabelas de resultados estão disponíveis como apêndice desta tese, acessíveis para esclarecimentos adicionais. Cada uma das quatro tabelas discutidas é acompanhada por gráficos, facilitando a visualização e compreensão dos dados.

O objetivo deste experimento é ilustrar a capacidade do simulador de auxiliar procedimentos de análise, envolvendo grande variação de aplicações, algoritmos e parâmetros de configuração. O objetivo específico é identificar as opções de mapeamento mais eficientes, dada uma estratégia de roteamento fixa, e como isso pode influenciar o desempenho de um sistema *many-core* baseado em NoCs no que diz respeito ao consumo de energia.

Analisando a Tabela 12 (configuração 6), observa-se um aumento no consumo de energia à medida que o número de tarefas nos grafos cresce. As configurações adotadas para os algoritmos, como *GA* (mutação por troca, 0.05, 5), *ACO* (0.1, 3, 1), PSO (0.4, 2.5, 1.5), TS (número de tarefas * 1.5, 20, 5) e SA (Tajary et al., 2022) (100, 0.03, 0.85), influenciaram os resultados obtidos.

O algoritmo GA, com sua estratégia evolutiva, destaca-se nas simulações para os Grafos$_{9}$, $_{25}$, $_{36}$, $_{49}$ e $_{100}$, obtendo o menor consumo de energia com valores de 350.00, 946.00, 1668.00, 2056.00 e 7576.00, respectivamente. Tais resultados enfatizam a eficácia do GA em se adaptar a diferentes escalas de problemas, utilizando suas capacidades genéticas para navegar e otimizar o espaço de buscas.

Por outro lado, o algoritmo SA (Tajary et al., 2022), beneficiando-se de uma estratégia de perturbação e atuando diretamente sobre as soluções do problema, permitiu a exploração do espaço de soluções de forma a evitar o aprisionamento

Tabela 12 – Desempenho dos Algoritmos de Mapeamento (configuração 6)

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et al.,2022)**
Grafo$_9$	**Média**	356.00	570.00	473.00	357.00	350.00
	Desvio Padrão	10.00	74.00	97.00	13.00	0.00
	Melhor	**350.00**	437.00	**350.00**	**350.00**	**350.00**
Grafo$_{16}$	**Média**	561.00	1187.00	1137.00	644.00	483.00
	Desvio Padrão	44.00	151.00	143.00	48.00	5.00
	Melhor	487.00	890.00	901.00	531.00	**473.00**
Grafo$_{25}$	**Média**	1092.00	2515.00	2492.00	1425.00	1002.00
	Desvio Padrão	79.00	211.00	251.00	89.00	27.00
	Melhor	946.00	1922.00	2054.00	1182.00	**907.00**
Grafo$_{36}$	**Média**	1896.00	4684.00	4614.00	2738.00	1892.00
	Desvio Padrão	101.00	378.00	371.00	143.00	46.00
	Melhor	**1668.00**	4014.00	3954.00	2392.00	1756.00
Grafo$_{49}$	**Média**	2255.00	6140.00	6055.00	3527.00	2398.00
	Desvio Padrão	91.00	466.00	369.00	200.00	47.00
	Melhor	**2056.00**	5369.00	4927.00	3001.00	2271.00
Grafo$_{64}$	**Média**	6001.00	10618.00	10100.00	6045.00	4024.00
	Desvio Padrão	233.00	573.00	628.00	357.00	89.00
	Melhor	5496.00	9508.00	8999.00	5195.00	**3763.00**
Grafo$_{81}$	**Média**	5307.00	13722.00	13247.00	7906.00	5121.00
	Desvio Padrão	237.00	867.00	802.00	437.00	109.00
	Melhor	4894.00	11909.00	11708.00	7114.00	**4747.00**
Grafo$_{100}$	**Média**	8766.00	20841.00	20244.00	12357.00	8080.00
	Desvio Padrão	375.00	1190.00	988.00	686.00	154.00
	Melhor	**7576.00**	18368.00	17336.00	10491.00	7621.00

em mínimos locais e potencialmente alcançar o mínimo global. O SA demonstrou eficiência semelhante ao GA para o Grafo$_9$, alcançando o valor de 350.00, e superou o GA para o Grafo$_{16}$, com um consumo de energia de apenas 473.00. Mostrou também excelente desempenho para os Grafos$_{25}$, $_{64}$ e $_{81}$, com valores de 907.00, 2271.00 e 3673.00, respectivamente. O desempenho significativo desse algoritmo em estruturas de problemas complexos ilustra sua robustez e escalabilidade diante do aumento da dificuldade e da quantidade de tarefas.

A representação gráfica desse desempenho, visualizada através de um gráfico de barras mostrado na Figura 5.9, oferece uma compreensão mais clara e imediata das diferenças no consumo de energia possibilitadas pelos algoritmos. As cores atribuídas a cada barra, correspondendo a cada algoritmo, facilitam a distinção visual e destacam tanto as áreas onde o GA é mais eficiente quanto aquelas em que o SA (Tajary et al., 2022) leva vantagem, proporcionando uma ilustração direta da eficácia de cada método em relação ao consumo de energia nos diferentes grafos analisados.

Ao examinar os novos resultados da Tabela 13 (configuração 9), que utiliza as seguintes combinações de parâmetros atualizadas dos algoritmos: GA (inserção,

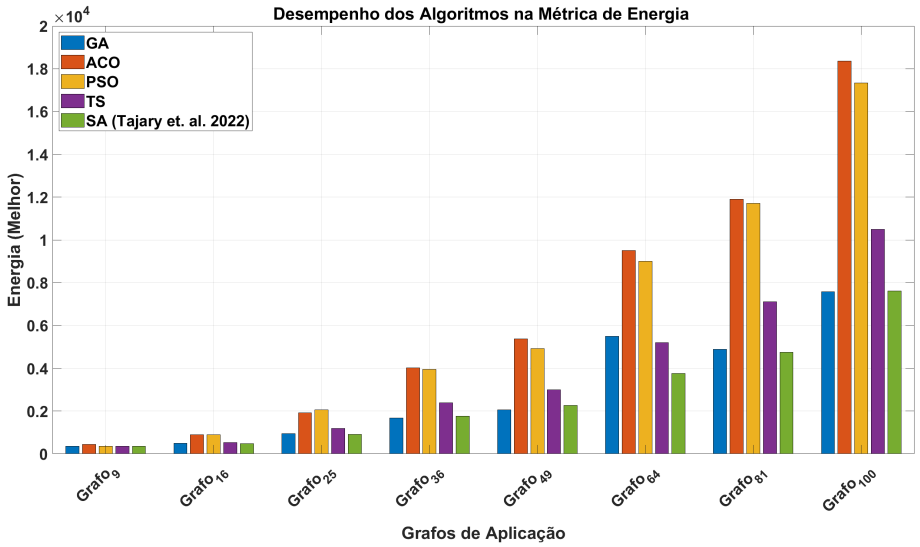

Figura 5.9 – Gráfico de Comparação de Mapeamento (configuração 6)

0.02, 5), ACO (0.5, 1, 1), PSO (0.6, 1.5, 1.5), TS (número de tarefas * 1.5, 10, 5), SA (Tajary et al., 2022) (150, 0.01, 0.85), foi possível notar a competição contínua entre GA e SA (Tajary et al., 2022) pelos melhores valores de desempenho nos diferentes grafos de aplicação. Para o Grafo_9, a igualdade de desempenho retorna, com GA, PSO, TS e SA (Tajary et al., 2022) todos atingindo o marco de eficiência energética de 350.00. No Grafo_{16}, o SA (Tajary et al., 2022) demonstra um consumo mínimo de energia de 473.00, o que já poderia ser esperado por ter apenas 16 tarefas no grafo.

Na sequência, os Grafos_{25}, $_{36}$ e $_{49}$ mostram o GA em melhores condições, com os menores consumos de energia registrados em 906.00, 1683.00 e 2084.00, respectivamente. Esses valores indicam a capacidade do GA, ao adotar a estratégia de inserção e uma menor taxa de mutação, de se adaptar de maneira otimizada à complexidade de valores médios de tarefas.

Nos Grafos_{64} e $_{100}$, no entanto, o SA (Tajary et al., 2022) retoma a liderança, com valores de 3673.00 e 7550.00. Aqui, a robustez do SA modificado em lidar com o aumento exponencial na complexidade das tarefas é evidente, sugerindo que, mesmo com parâmetros alterados, continuou reagindo de forma eficiente em cenários com maior número de tarefas.

Em contraste, no Grafo_{81}, o GA volta a se destacar com um valor de 4691.00,

Tabela 13 – Desempenho dos Algoritmos de Mapeamento (configuração 9)

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. (2022)**
Grafo$_9$	**Média**	353.00	571.00	496.00	353.00	350.00
	Desvio Padrão	7.00	79.00	87.00	7.00	0.00
	Melhor	**350.00**	409.00	**350.00**	**350.00**	**350.00**
Grafo$_{16}$	**Média**	554.00	1221.00	1193.00	574.00	479.00
	Desvio Padrão	39.00	145.00	156.00	34.00	5.00
	Melhor	487.00	914.00	858.00	486.00	**473.00**
Grafo$_{25}$	**Média**	1048.00	2591.00	2435.00	1231.00	992.00
	Desvio Padrão	66.00	237.00	258.00	78.00	22.00
	Melhor	**906.00**	2111.00	1899.00	1067.00	923.00
Grafo$_{36}$	**Média**	1867.00	4749.00	4661.00	2299.00	1863.00
	Desvio Padrão	98.00	363.00	335.00	123.00	41.00
	Melhor	**1683.00**	3840.00	3997.00	2021.00	1738.00
Grafo$_{49}$	**Média**	2294.00	6224.00	6174.00	2961.00	2362.00
	Desvio Padrão	104.00	342.00	416.00	179.00	55.00
	Melhor	**2084.00**	5531.00	5115.00	2646.00	2147.00
Grafo$_{64}$	**Média**	5970.00	10572.00	10105.00	4924.00	3968.00
	Desvio Padrão	255.00	676.00	620.00	288.00	91.00
	Melhor	5418.00	9074.00	8482.00	4316.00	**3673.00**
Grafo$_{81}$	**Média**	5371.00	13584.00	13264.00	6367.00	5056.00
	Desvio Padrão	259.00	685.00	748.00	335.00	94.00
	Melhor	**4691.00**	11988.00	11251.00	5651.00	4779.00
Grafo$_{100}$	**Média**	8840.00	21050.00	20429.00	10125.00	7963.00
	Desvio Padrão	369.00	966.00	1101.00	544.00	145.00
	Melhor	8128.00	18756.00	17588.00	9040.00	**7550.00**

sublinhando a dinâmica contínua de competição entre os dois algoritmos ao longo de variados graus de complexidade do problema.

O padrão de alternância entre o GA e o SA (Tajary et al., 2022) como os melhores em diferentes configurações de grafo enfatiza a importância da seleção e configuração cuidadosa de algoritmos em problemas de otimização de energia. Conforme visualizado em um gráfico de barras comparativo, esses resultados se traduzem em uma representação gráfica que ilustra uma contínua disputa pela eficiência, com GA e SA alternando posições de superioridade dependendo do tamanho e da complexidade dos grafos, refletindo suas respectivas vantagens adaptativas e estratégias de otimização.

Ao revisar os resultados atualizados na Tabela 14 (configuração 12) com os parâmetros recém-ajustados dos algoritmos: GA (inserção, 0.01, 5), ACO (0.5, 2, 1), PSO (0.6, 2.0, 1.5), TS (número de tarefas * 1.5, 20, 5) e SA (Tajary et al., 2022) (150, 0.02, 0.85), observa-se uma dinâmica interessante no desempenho entre os algoritmos, particularmente entre o GA e o SA (Tajary et al., 2022). Essa análise revela uma dominância notável do SA (Tajary et al., 2022) em quase todos os grafos, exceto no Grafo$_9$, onde GA, TS e SA (Tajary et al., 2022) compartilham

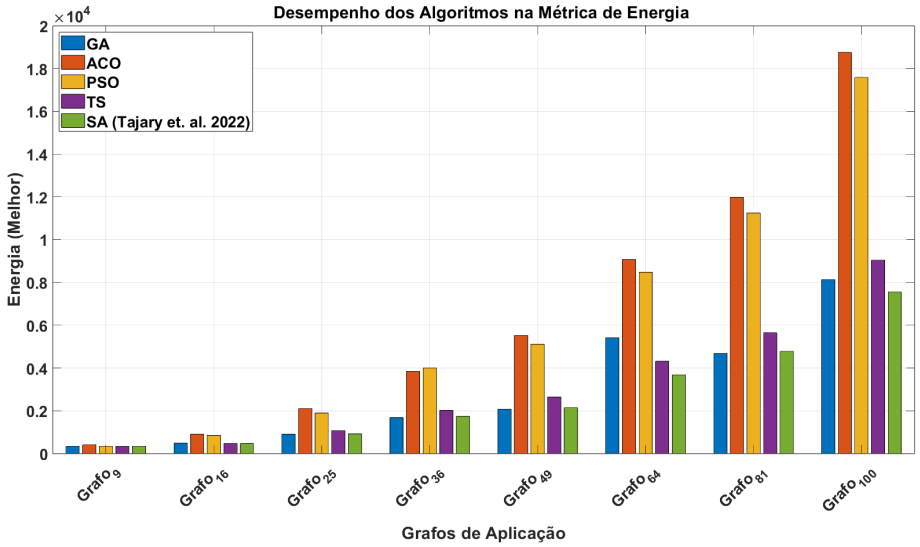

Figura 5.10 – Gráfico de Comparação de Mapeamento (configuração 9)

o melhor valor de eficiência energética de 350.00.

A partir do Grafo_{16}, o SA (Tajary et al., 2022) supera consistentemente todos os outros algoritmos, alcançando os melhores valores de desempenho em todos os grafos subsequentes. Os valores específicos de desempenho para o SA (Tajary et al., 2022) são 473.00 para o Grafo_{16}, 936.00 para o Grafo_{25}, e continuam melhorando em complexidade e número de tarefas, culminando em 7707.00 para o Grafo_{100}.

Esses resultados indicam que as adaptações feitas nos parâmetros do SA (Tajary et al., 2022), incluindo uma ligeira alteração nos critérios de resfriamento, proporcionaram uma vantagem significativa na otimização do consumo de energia em cenários de alta complexidade. Isso sugere que o método de perturbação adaptado do SA, combinado com uma abordagem de resfriamento cuidadosamente calibrada, é particularmente eficaz em espaços de solução maiores e mais desafiadores, característicos de grafos com um número elevado de tarefas.

Visualizado através de um gráfico de barras, ilustrado na Figura 5.11, essa tendência de desempenho mostra a escalabilidade e eficiência do SA (Tajary et al., 2022) em comparação com outras abordagens. As barras associadas ao SA, destacadas por uma cor distinta, demonstram uma altura consistentemente menor, simbolizando a otimização superior de energia através de uma ampla gama de complexidades de grafo. Esse fato não apenas reforça a importância da seleção e

Tabela 14 – Desempenho dos Algoritmos de Mapeamento (configuração 12)

Grafos de Aplicação	Métricas	Adaptado do PlatEMO			Implementado	Implementado
		GA	ACO	PSO	TS	SA (Tajary et al.,2022)
$\mathbf{Grafo_9}$	**Média**	364.00	579.00	539.00	359.00	350.00
	Desvio Padrão	18.00	72.00	102.00	13.00	0.00
	Melhor	**350.00**	402.00	351.00	**350.00**	**350.00**
$\mathbf{Grafo_{16}}$	**Média**	641.00	1186.00	1137.00	666.00	486.00
	Desvio Padrão	65.00	158.00	126.00	48.00	6.00
	Melhor	515.00	819.00	852.00	559.00	**473.00**
$\mathbf{Grafo_{25}}$	**Média**	1344.00	2636.00	2494.00	1466.00	1015.00
	Desvio Padrão	124.00	265.00	189.00	90.00	28.00
	Melhor	1141.00	2088.00	2054.00	1264.00	**936.00**
$\mathbf{Grafo_{36}}$	**Média**	2492.00	4655.00	4676.00	2787.00	1914.00
	Desvio Padrão	193.00	385.00	323.00	187.00	51.00
	Melhor	2079.00	3735.00	3998.00	2381.00	**1794.00**
$\mathbf{Grafo_{49}}$	**Média**	3306.00	6260.00	6029.00	3620.00	2435.00
	Desvio Padrão	256.00	415.00	387.00	244.00	54.00
	Melhor	2860.00	5380.00	5052.00	3038.00	**2288.00**
$\mathbf{Grafo_{64}}$	**Média**	5759.00	10431.00	10119.00	6142.00	4067.00
	Desvio Padrão	365.00	575.00	641.00	328.00	101.00
	Melhor	4851.00	9369.00	8659.00	5333.00	**3743.00**
$\mathbf{Grafo_{81}}$	**Média**	7855.00	13839.00	13231.00	8068.00	5193.00
	Desvio Padrão	508.00	797.00	618.00	585.00	120.00
	Melhor	6544.00	12006.00	12058.00	6825.00	**4885.00**
$\mathbf{Grafo_{100}}$	**Média**	12725.00	20887.00	20548.00	12783.00	8177.00
	Desvio Padrão	772.00	897.00	1173.00	744.00	158.00
	Melhor	10957.00	18921.00	17919.00	10907.00	**7707.00**

ajuste de parâmetros em algoritmos de otimização, mas também destaca a robustez e flexibilidade do SA (Tajary et al., 2022) diante do aumento da complexidade das tarefas.

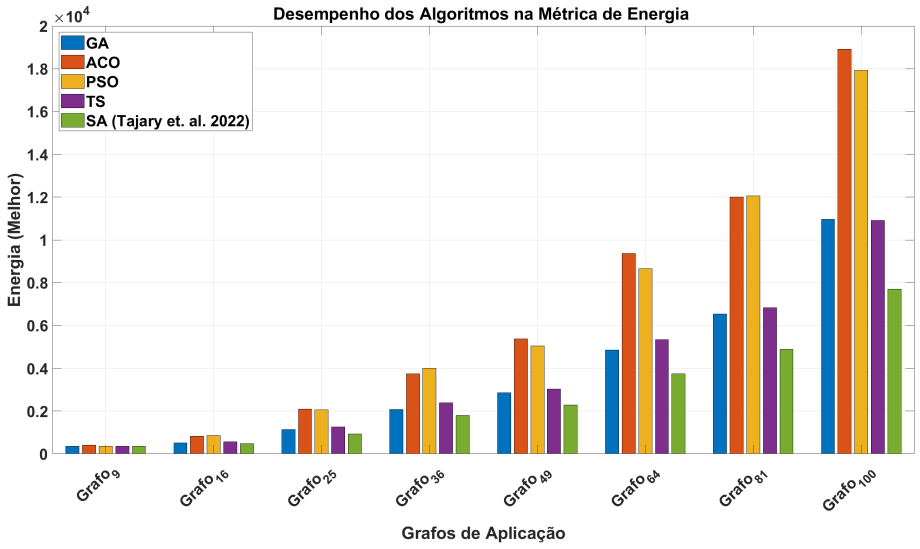

Figura 5.11 – Gráfico de Comparação de Mapeamento (configuração 12)

Ao adaptar os algoritmos com os parâmetros mais recentes: GA (inversão, 0.1, 5), ACO (0.9, 2, 1), PSO (0.8, 2.0, 1.5), TS (número de tarefas * 1.5, 20, 5) e SA (Tajary et al., 2022) (200, 0.02, 0.85), para os resultados apresentados na Tabela 15 (configuração 21), nota-se uma mudança significativa no desempenho dos algoritmos. Esse conjunto de resultados mostra uma predominância clara do GA em todos os grafos, atribuída principalmente à sua alta taxa de mutação de 0.1, que parece ter sido um fator decisivo para seu sucesso.

No Grafo_9, como em todas as tabelas e variações, o GA, TS e SA (Tajary et al., 2022) alcançam o melhor valor de 350.00, indicando uma otimização energética eficaz. No entanto, à medida que a complexidade dos grafos aumenta, o GA destaca-se exclusivamente como o algoritmo superior. Desde o Grafo_{16}, com um desempenho de 517.00, até o mais complexo Grafo_{100}, onde atinge um valor de 11167.00, o GA demonstra uma capacidade excepcional de adaptar-se e otimizar em ambientes cada vez mais complexos.

Essa superioridade do GA em todos os estágios sugere que a estratégia de inversão, combinada com a taxa de mutação elevada, oferece uma vantagem única na exploração do espaço de soluções, permitindo uma navegação eficaz através de problemas complexos. Os outros algoritmos, apesar de suas configurações ajustadas, não conseguiram igualar sua eficácia, possivelmente devido às limitações em suas estratégias de busca ou na sensibilidade às suas respectivas taxas de parâmetros ajustados.

A visualização dessa tendência em um gráfico de barras (Figura 5.12) permite observar a superioridade consistente do GA em relação aos outros algoritmos, destacando-se com barras mais baixas em cenários de todas as complexidades. Esse padrão ressalta a importância da seleção de parâmetros na otimização de algoritmos, especialmente em aplicações onde a eficiência energética é crítica. O ajuste fino da taxa de mutação do GA, em particular, revelou-se uma abordagem capaz de alcançar resultados significativos em uma ampla gama de cenários de teste, reafirmando o potencial do GA como uma ferramenta robusta e versátil para desafios de otimização de energia nos cenários abordados.

A discrepância nas métricas de desempenho entre distintas configurações de parâmetros mostrou-se mínima, sugerindo que ajustes profundos nos parâmetros impactam pouco o resultado. Isso indica que é possível empregar valores padrão ou

Tabela 15 – Desempenho dos Algoritmos de Mapeamento (configuração 21)

Grafos de Aplicação	Métricas	Adaptado do PlatEMO			Implementado	Implementado
		GA	ACO	PSO	TS	SA (Tajary et al.,2022)
$Grafo_9$	Média	359.00	591.00	555.00	360.00	360.00
	Desvio Padrão	14.00	74.00	72.00	13.00	13.00
	Melhor	**350.00**	402.00	420.00	**350.00**	**350.00**
$Grafo_{16}$	Média	632.00	1196.00	1148.00	661.00	661.00
	Desvio Padrão	52.00	146.00	154.00	48.00	48.00
	Melhor	**517.00**	859.00	822.00	539.00	539.00
$Grafo_{25}$	Média	1334.00	2553.00	2519.00	1477.00	1477.00
	Desvio Padrão	105.00	206.00	233.00	106.00	106.00
	Melhor	**1109.00**	2217.00	2020.00	1133.00	1133.00
$Grafo_{36}$	Média	2543.00	4831.00	4686.00	2812.00	2812.00
	Desvio Padrão	204.00	357.00	329.00	172.00	172.00
	Melhor	**2183.00**	3998.00	3891.00	2371.00	2371.00
$Grafo_{49}$	Média	3350.00	6250.00	6141.00	3640.00	3640.00
	Desvio Padrão	207.00	398.00	426.00	242.00	242.00
	Melhor	**2841.00**	5073.00	5156.00	3158.00	3158.00
$Grafo_{64}$	Média	5867.00	10510.00	10282.00	6190.00	6190.00
	Desvio Padrão	340.00	618.00	502.00	392.00	392.00
	Melhor	**5053.00**	8765.00	9396.00	5361.00	5361.00
$Grafo_{81}$	Média	7856.00	13447.00	13303.00	8118.00	8118.00
	Desvio Padrão	568.00	737.00	880.00	495.00	495.00
	Melhor	**6605.00**	11882.00	11579.00	6975.00	6975.00
$Grafo_{100}$	Média	13034.00	20959.00	20404.00	12735.00	12735.00
	Desvio Padrão	635.00	842.00	1179.00	735.00	735.00
	Melhor	**11167.00**	19309.00	17962.00	11203.00	11203.00

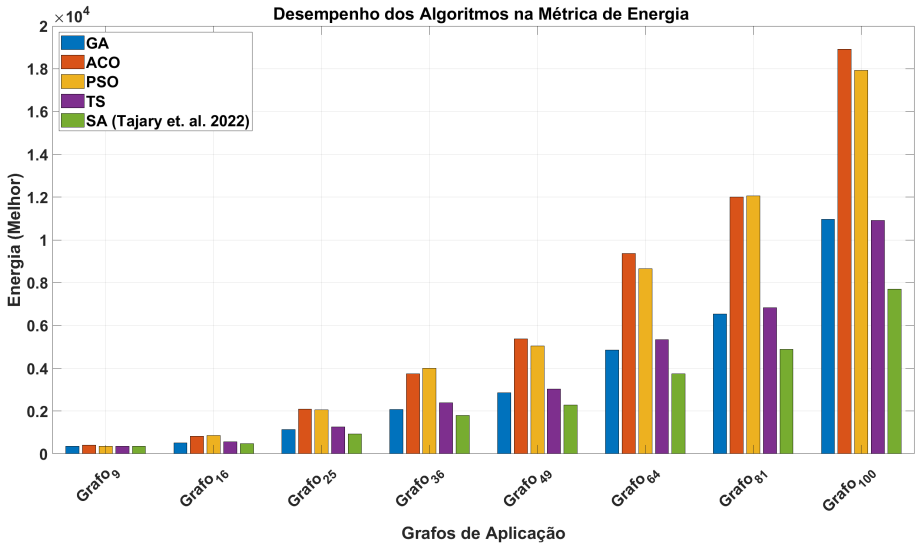

Figura 5.12 – Gráfico de Comparação de Mapeamento (configuração 21)

realizar leves ajustes, focando na dimensão do grafo em vez da complexidade dos parâmetros. A superioridade dos algoritmos reflete sua aptidão em se adaptar à complexidade dos grafos, promovendo uma metodologia simplificada para a escolha de algoritmos de mapeamento. Esta abordagem prioriza a análise da estrutura do

grafo, ao invés de testes extensivos com diversas configurações de parâmetros.

5.4 Experimento de DSE com Várias Estratégias para Roteamento e Única para Mapeamento

Nesta seção, são apresentados experimentos que analisam o consumo de energia ao comparar diferentes algoritmos de roteamento, utilizando o GA como método de mapeamento fixo. Deve-se notar que, apesar de o GA ser o único algoritmo de mapeamento adotado para tal experimento, foram consideradas 9 variações de seus parâmetros específicos, gerando 9 tabelas de resultados, das quais 3 são apresentadas e discutidas nesta seção, enquanto as outras 6 estão disponibilizadas no apêndice para informações adicionais.

Essa análise visa identificar a relação entre a escolha de algoritmos de roteamento e seu impacto no consumo de energia, contribuindo para avaliar e aprimorar a estrutura do simulador em relação às abordagens disponíveis em sua base de dados. Isso garante que o usuário possa implementar ou integrar outros algoritmos de sua preferência. Por meio dessas variações e da implementação do GA, busca-se esclarecer os fatores que afetam a eficiência energética em arquiteturas de *many-core* baseadas em NoCs.

Os primeiros dados coletados para a comparação foram baseados na "configuração 0"de parâmetros, ou seja, mutação por troca com uma taxa de mutação de 0.1. Os resultados estão descritos na Tabela 16 e ilustrados no gráfico da Figura 5.13. Nos grafos de menor complexidade, Grafo_9 e Grafo_{16}, as diferenças entre os algoritmos são mínimas, com o algoritmo *Negative-First* se destacando pelo menor consumo energético. Entretanto, à medida que a complexidade dos grafos aumenta, essas diferenças tornam-se mais evidentes.

Nos Grafo_{25}, os algoritmos XY e XYX mostram os melhores resultados, similarmente ao *North-Last* no Grafo_{36}. Com o avanço para o Grafo_{49}, os algoritmos determinísticos XY e XYX apresentam novamente os melhores resultados, e o XY também se sobressai no Grafo_{81}, sugerindo que podem ser preferenciais em contextos que valorizam a previsibilidade.

No entanto, o desempenho dos algoritmos adaptativos não deve ser negligen-

Tabela 16 – Desempenho de Algoritmos de Roteamento (configuração 0)

Grafos	Estatísticas	West-First	Negative-First	North-Last	Odd-Even	XY	XYX
Grafo$_9$	Média	89,00	96,00	102,00	97,00	103,00	103,00
	Desvio Padrão	34,00	42,00	38,00	35,00	34,00	34,00
	Melhor	19,00	**18,00**	37,00	47,00	37,00	37,00
Grafo$_{16}$	Média	324,00	312,00	319,00	313,00	317,00	317,00
	Desvio Padrão	56,00	56,00	70,00	64,00	68,00	68,00
	Melhor	206,00	**172,00**	186,00	180,00	213,00	213,00
Grafo$_{25}$	Média	773,00	747,00	767,00	756,00	770,00	766,00
	Desvio Padrão	121,00	72,00	120,00	113,00	117,00	114,00
	Melhor	439,00	612,00	505,00	517,00	**433,00**	**433,00**
Grafo$_{36}$	Média	1497,00	1489,00	1470,00	1478,00	1487,00	1480,00
	Desvio Padrão	148,00	130,00	141,00	158,00	157,00	166,00
	Melhor	1136,00	1156,00	**1036,00**	1049,00	1216,00	1073,00
Grafo$_{49}$	Média	1872,00	1883,00	1901,00	1879,00	1886,00	1873,00
	Desvio Padrão	149,00	146,00	160,00	135,00	151,00	163,00
	Melhor	1603,00	1620,00	1575,00	1554,00	**1443,00**	**1443,00**
Grafo$_{64}$	Média	3311,00	3313,00	3350,00	3328,00	3347,00	3339,00
	Desvio Padrão	250,00	215,00	259,00	207,00	216,00	238,00
	Melhor	2642,00	**2580,00**	2853,00	2864,00	2877,00	2846,00
Grafo$_{81}$	Média	4467,00	4485,00	4475,00	4485,00	4462,00	4459,00
	Desvio Padrão	291,00	304,00	280,00	236,00	240,00	224,00
	Melhor	3770,00	3781,00	3841,00	3934,00	**3747,00**	3905,00
Grafo$_{100}$	Média	7524,00	7447,00	7532,00	7488,00	7430,00	7412,00
	Desvio Padrão	410,00	422,00	366,00	379,00	353,00	376,00
	Melhor	6497,00	**6367,00**	6509,00	6595,00	6856,00	6416,00

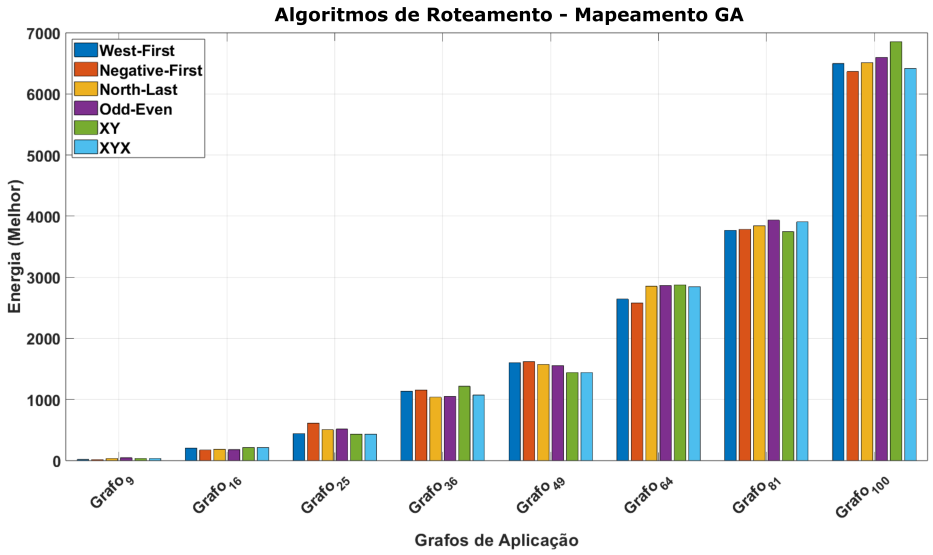

Figura 5.13 – Gráfico de Comparação de Roteamento (configuração 0)

ciado. Em cenários de alta complexidade, como no Grafo$_{100}$, o *Negative-First* demonstra novamente sua eficiência, indicando sua capacidade de adaptação a

condições mais desafiadoras.

A análise subsequente explora a eficiência energética de algoritmos de roteamento sob uma nova configuração de parâmetros para GA (configuração 4): mutação por inserção com uma taxa de mutação de 0.02. Os resultados, detalhados na Tabela 17 e visualizados no gráfico da Figura 5.14, revelam variações no desempenho em comparação com a configuração anterior (Tabela 16). No Grafo_9, o algoritmo *Odd-Even*, anteriormente não destacado, surgiu como o mais eficiente, registrando o menor consumo energético. Para o Grafo_{16}, o *West-First* mostrou uma melhoria significativa, alcançando o melhor desempenho, indicando a sensibilidade do algoritmo a mudanças nos parâmetros de mutação do GA.

Tabela 17 – Desempenho de Algoritmos de Roteamento (configuração 4)

Grafos	Estatísticas	West-First	Negative-First	North-Last	Odd-Even	XY	XYX
Grafo_9	Média	108,00	114,00	113,00	102,00	110,00	110,00
	Desvio Padrão	45,00	42,00	41,00	39,00	39,00	39,00
	Melhor	37,00	37,00	98,00	**19,00**	47,00	47,00
Grafo_{16}	Média	310,00	321,00	334,00	313,00	313,00	313,00
	Desvio Padrão	78,00	53,00	64,00	72,00	65,00	65,00
	Melhor	**111,00**	201,00	196,00	185,00	184,00	184,00
Grafo_{25}	Média	802,00	783,00	808,00	820,00	787,00	790,00
	Desvio Padrão	110,00	92,00	99,00	104,00	116,00	113,00
	Melhor	570,00	553,00	557,00	614,00	**537,00**	**537,00**
Grafo_{36}	Média	1644,00	1618,00	1639,00	1623,00	1632,00	1643,00
	Desvio Padrão	180,00	160,00	152,00	158,00	162,00	147,00
	Melhor	1279,00	1283,00	1342,00	**1189,00**	1237,00	1237,00
Grafo_{49}	Média	2254,00	2314,00	2276,00	2337,00	2250,00	2254,00
	Desvio Padrão	170,00	167,00	157,00	193,00	174,00	193,00
	Melhor	1928,00	**1748,00**	1969,00	1816,00	1808,00	1806,00
Grafo_{64}	Média	4245,00	4156,00	4218,00	4195,00	4198,00	4121,00
	Desvio Padrão	217,00	222,00	307,00	246,00	294,00	315,00
	Melhor	3881,00	**3484,00**	3486,00	3654,00	3595,00	3580,00
Grafo_{81}	Média	5778,00	5720,00	5749,00	5808,00	5796,00	5755,00
	Desvio Padrão	330,00	292,00	284,00	326,00	358,00	385,00
	Melhor	**4798,00**	5243,00	5182,00	4968,00	5034,00	5105,00
Grafo_{100}	Média	9698,00	9574,00	9481,00	9561,00	9661,00	9622,00
	Desvio Padrão	513,00	442,00	537,00	456,00	492,00	411,00
	Melhor	8570,00	8672,00	8527,00	8744,00	**8265,00**	8717,00

Nos grafos de maior escala, o desempenho dos algoritmos XY e XYX, que foram previamente eficientes, continua mostrando sua eficiência. No entanto, o *Negative-First*, particularmente no Grafo_{64}, e o *West-First* no Grafo_{81}, apresentaram o melhor consumo de energia. Esse cenário demonstra uma variação de desempenho que sugere uma influência significativa dos parâmetros do GA na eficiência dos algoritmos de roteamento.

No Grafo_{100}, a configuração favoreceu o algoritmo XY novamente, destacando-se como o mais eficiente dentre os testados. Comparando com os dados anteriores, onde o *Negative-First* e os algoritmos determinísticos tinham um desempenho superior, observa-se que pequenas alterações nos parâmetros do GA podem resultar em mudanças expressivas na hierarquia de eficiência energética dos algoritmos de roteamento. Isso sublinha a importância de uma análise abrangente das configurações do GA ao se avaliar o roteamento em NoCs.

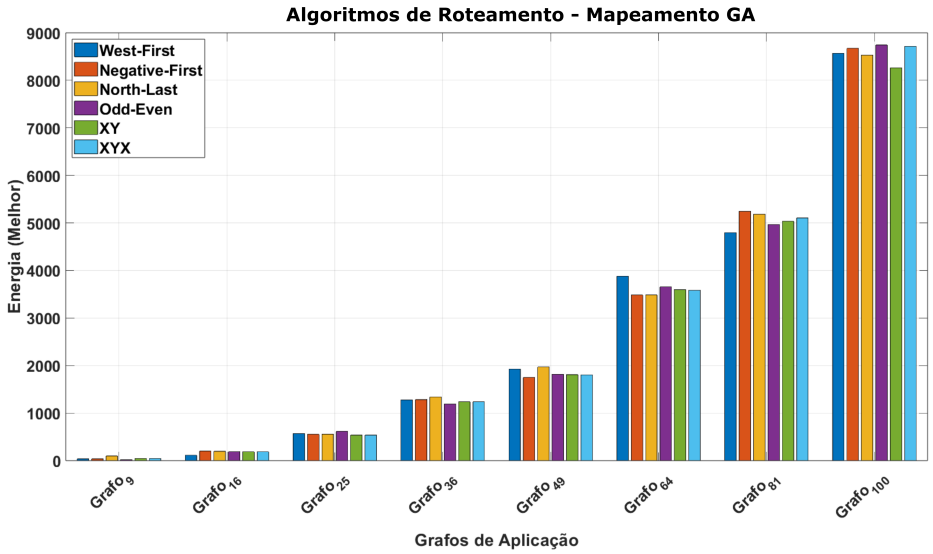

Figura 5.14 – Gráfico de Comparação de Roteamento (configuração 4)

A configuração final apresentada (configuração 8) abrange a mutação por inversão com uma taxa de mutação de 0.05, com o intuito de examinar seu impacto no consumo energético dos algoritmos de roteamento. Os dados pertinentes são apresentados na Tabela 18 e visualmente no gráfico da Figura 5.15. Para o Grafo_9, o *West-First* apresenta o menor consumo energético, significativamente melhor do que o observado em configurações anteriores. Esta mudança pode sugerir uma adaptação favorável desse algoritmo ao aumento da taxa de mutação.

Em relação ao Grafo_{16}, enquanto a média se manteve consistente entre os algoritmos, o *West-First* destaca-se novamente com o menor consumo energético, mostrando uma tendência de eficiência energética sob esta configuração de mutação. Avançando para o Grafo_{25}, observa-se uma inversão na tendência anterior,

com o XY alcançando o menor consumo, refletindo um possível alinhamento ótimo entre o algoritmo e os parâmetros de mutação.

No contexto do Grafo_{36}, o *West-First* mantém uma posição de destaque em eficiência energética, reforçando a ideia de que tal algoritmo é particularmente sensível a essa taxa de mutação específica. Já no Grafo_{49}, é o *North-Last* que emerge como o mais eficiente, substituindo os algoritmos que previamente dominavam essa faixa de complexidade.

Nos grafos de maior escala, como os de 64 e 81 tarefas, o desempenho energético melhora em relação a outras configurações de mutação. Especificamente, no Grafo_{64}, os algoritmos XY e XYX compartilham o melhor resultado, e no Grafo_{81}, o XY é o mais eficiente. Esses resultados evidenciam uma resposta positiva desses algoritmos a uma taxa de mutação mais elevada.

Tabela 18 – Desempenho de Algoritmos de Roteamento (configuração 8)

Grafos	Estatísticas	West-First	Negative-First	North-Last	Odd-Even	XY	XYX
Grafo_9	Média	102,00	101,00	105,00	102,00	108,00	108,00
	Desvio Padrão	35,00	38,00	30,00	37,00	36,00	36,00
	Melhor	**18,00**	37,00	47,00	19,00	37,00	37,00
Grafo_{16}	Média	315,00	318,00	338,00	318,00	312,00	311,00
	Desvio Padrão	71,00	66,00	57,00	59,00	59,00	58,00
	Melhor	**165,00**	193,00	212,00	199,00	193,00	193,00
Grafo_{25}	Média	776,00	786,00	795,00	806,00	748,00	769,00
	Desvio Padrão	103,00	87,00	97,00	119,00	106,00	121,00
	Melhor	543,00	640,00	637,00	582,00	**526,00**	**526,00**
Grafo_{36}	Média	1514,00	1535,00	1489,00	1493,00	1528,00	1536,00
	Desvio Padrão	150,00	152,00	158,00	146,00	166,00	144,00
	Melhor	**1035,00**	1288,00	1176,00	1204,00	1172,00	1175,00
Grafo_{49}	Média	1921,00	2015,00	2021,00	1994,00	2001,00	2016,00
	Desvio Padrão	164,00	138,00	188,00	163,00	150,00	142,00
	Melhor	1556,00	1745,00	**1470,00**	1659,00	1667,00	1667,00
Grafo_{64}	Média	3574,00	3690,00	3628,00	3701,00	3631,00	3624,00
	Desvio Padrão	203,00	230,00	241,00	250,00	194,00	231,00
	Melhor	3017,00	3139,00	3029,00	3147,00	**3078,00**	**3078,00**
Grafo_{81}	Média	4955,00	4936,00	5036,00	4994,00	4904,00	4963,00
	Desvio Padrão	312,00	344,00	322,00	299,00	248,00	245,00
	Melhor	4260,00	4272,00	4400,00	4273,00	**4235,00**	4301,00
Grafo_{100}	Média	8354,00	8259,00	8240,00	8247,00	8275,00	8270,00
	Desvio Padrão	438,00	429,00	339,00	361,00	385,00	365,00
	Melhor	7323,00	7389,00	7479,00	7659,00	**7084,00**	7388,00

Finalmente, para o Grafo_{100}, o algoritmo XY sobressai-se com o menor consumo de energia, um resultado que destaca a consistência desse algoritmo mesmo em tarefas de alta complexidade. Comparado com a tabela anterior, verifica-se que a mutação por inversão e a taxa de mutação de 0.05 modificam significativamente

o desempenho dos algoritmos, enfatizando a necessidade de avaliações contextuais para a escolha da configuração de mutação no GA.

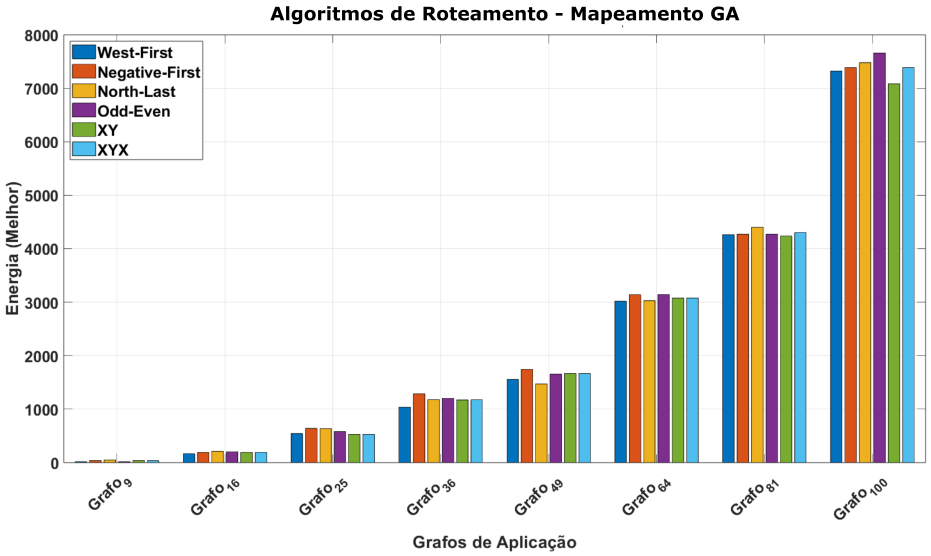

Figura 5.15 – Gráfico de Comparação de Roteamento (configuração 8)

Este modelo de experimentos revelou que, mesmo com o aumento da complexidade das tarefas, a alteração dos parâmetros do GA influencia diretamente a competição entre as estratégias de roteamento. Este fenômeno ocorre sem a necessidade de repetições extensivas. Apesar dos resultados próximos entre si, a análise sugere que os algoritmos, tanto determinísticos quanto adaptativos, podem ser avaliados de forma mais precisa dentro do simulador, oferecendo entendimentos sobre sua eficácia em diferentes cenários.

5.5 Considerações finais

Neste capítulo, foram consolidados os resultados obtidos nos experimentos, validando a confiabilidade do simulador desenvolvido e sua adequação para experimentos de DSE, analisando a eficácia de algoritmos de mapeamento e roteamento em NoCs quanto à otimização do consumo de energia. Os testes ressaltaram a performance consistente dos algoritmos de mapeamento GA e SA (Tajary, et al., 2022), indicando a viabilidade de utilizar parâmetros padrão ou ajustes mínimos, de acordo com a complexidade da aplicação representada por um DAG. Identificou-se também que as estratégias de roteamento são influenciadas tanto pela complexidade das tarefas quanto pelos parâmetros do algoritmo de mapeamento GA. O simulador proposto mostrou-se adequado para realizar uma avaliação precisa da interação entre esses algoritmos e sua influência no desempenho de um sistema *many-core* baseado em NoCs.

Capítulo 6

Conclusões

Neste capítulo, é apresentada uma síntese do contexto, objetivos, desenvolvimento, resultados obtidos e possíveis extensões do projeto de pesquisa que deu origem a esta tese de doutorado.

A constatação das limitações de aumento de desempenho de microprocessadores com um único núcleo, observadas no início dos anos 2000, aliada aos avanços no projeto e implementação de *chips* resultantes da chamada *Lei de Moore*, motivou o desenvolvimento das arquiteturas *multi-core*, que evoluíram para estruturas de processamento *many-core*, podendo conter centenas de núcleos de processamento. Essas arquiteturas são utilizadas principalmente como plataformas de execução para aplicações com requisitos de alto desempenho, como computação avançada.

A natureza inerentemente paralela dos processadores *many-core* apresenta grande potencial para exploração de paralelismo em nível de tarefas. Porém, de maneira análoga aos multiprocessadores convencionais, a exploração eficiente dessas máquinas apresenta desafios na divisão e comunicação entre tarefas.

Por se constituírem em multiprocessadores de chip único, sistemas *many-core* oferecem novas possibilidades para a comunicação entre núcleos de processamento, sendo uma delas a chamada *Network-on-Chip*. Assim, abrem-se novas possibilidades para otimizar o uso de plataformas de execução *many-core* baseadas nessa abordagem, tanto no que diz respeito ao suporte de *hardware* quanto na dispo-

nibilidade de algoritmos para mapeamento de tarefas e roteamento de pacotes. Além dos algoritmos bem estabelecidos empregados para essas funções, há grande interesse na aplicação de várias alternativas de estratégias que têm surgido nos últimos anos, como os chamados meta-heurísticos.

Essas novas possibilidades apresentam grande potencial de ganho de desempenho, viabilizando novas aplicações computacionais. No entanto, também trazem desafios para a escolha da melhor solução de *hardware* e *software* que atenda aos objetivos de um projeto, devido à necessidade de considerar um grande número de opções interagindo entre si. Para lidar com esse problema, é necessário adotar e usar simuladores, tentando antecipar as vantagens e desvantagens de uma determinada opção de projeto. Nesse contexto, os simuladores para NoCs são usualmente adotados para auxiliar a análise e decisões sobre opções de projeto.

Após uma revisão de literatura, constatou-se a existência de uma diversidade de simuladores específicos para NoCs, cada um com suas particularidades e focos de análise. Como característica comum à grande maioria deles, observou-se a especialização para lidar com redes e algoritmos com características bem definidas e alto grau de detalhamento. Apesar de serem eficientes para as finalidades propostas, possuem pouca flexibilidade, limitando as possibilidades de experimentação em atividades de exploração do espaço de opções de projeto DSE.

Diante dessa constatação, o projeto de pesquisa desenvolvido e apresentado nesta tese teve como objetivo conceber e desenvolver um simulador de alto nível para arquiteturas *many-core* baseadas em NoCs, organizado em módulos. Ele deve estimar parâmetros de desempenho desses sistemas sem se ater a detalhes específicos de *hardware*. Visa ser mais adequado para estudos e comparações iniciais, incluindo algoritmos adaptados para o mapeamento de tarefas e roteamento de pacotes, no contexto de *DSE*. O objetivo é oferecer características que permitam aos usuários assimilar os conceitos por meio de estudos em um ambiente controlado e acessível. O simulador desenvolvido, chamado Simulador NoC, possui dois componentes principais em seu núcleo de processamento: a representação da arquitetura NoC e um repositório de algoritmos para estratégias de mapeamento e roteamento, interconectados por uma interface dedicada a converter e padronizar dados de simulação e funcionamento do simulador.

Com o objetivo de validar o método e a abordagem proposta em relação aos

objetivos estabelecidos para o projeto, uma série de experimentos foi conduzida. Inicialmente, buscou-se validar a confiabilidade do projeto como um todo, quanto aos resultados produzidos para a métrica de estimativa do consumo de energia. Usando como base um simulador mais realista, o NoCTweak, constatou-se que os resultados produzidos pelo Simulador NoC apresentam clara correlação com aqueles do simulador de referência, indicando confiabilidade no processamento e geração de dados. No que diz respeito ao uso do ambiente de simulação como suporte de experimentação e apoio à tomada de decisões de projeto, dois extensos experimentos foram conduzidos. Um deles explorou diversos algoritmos customizados para o mapeamento (variando parâmetros de configuração para cada um), adotando uma estratégia única para roteamento. O outro experimento adotou uma estratégia única de mapeamento, buscando explorar diversos algoritmos de roteamento.

Esses experimentos confirmaram a adequação do simulador para análise do tipo DSE, evidenciando diferenças significativas entre opções de projeto de acordo com a complexidade do DAG representando uma aplicação de interesse. Diante do exposto, e confrontando o simulador desenvolvido com os considerados na revisão de literatura, pode-se concluir que o método e a abordagem desenvolvidos atingiram os objetivos do projeto que deu origem a esta tese de doutorado.

Contudo, deve ser mencionado que este trabalho não está isento de limitações. A abstração em alto nível, embora facilitando a compreensão e a aplicabilidade dos conceitos, pode não capturar todas as complexidades de desempenho e eficiência energética presentes em simulações mais detalhadas e específicas. Além disso, a diversidade de algoritmos e métricas de desempenho focou primariamente no consumo de energia, podendo haver outros aspectos relevantes, como latência, *throughput* e confiabilidade, que não foram implementados nesta versão do simulador.

Diante dessas limitações e dos resultados obtidos, propõem-se para trabalhos futuros a expansão do simulador, incorporando métricas adicionais e a possibilidade de exploração de outras topologias para NoCs. A integração de novos algoritmos de mapeamento e roteamento, seja através da plataforma PlatEMO ou mediante implementações próprias, também se constitui em um aprimoramento natural do simulador, o que pode ser feito com relativa facilidade dada a sua

natureza modular.

Adicionalmente, o desenvolvimento de uma interface gráfica para o simulador enriquecerá sua usabilidade em atividades de ensino e pesquisa, tornando-o mais acessível e intuitivo. Tais desenvolvimentos não apenas superarão as limitações identificadas, mas também contribuirão significativamente para atividades de ensino e pesquisa nesse contexto.

Referências

ABBAS NOREEN JAMIL, M. U. S.; ABBAS, A. Congestion-aware routing algorithm for noc using data packets. **Wireless Communications and Mobile Computing**, v. 2021, p. 1–10, 2021. Disponível em: <https://www.hindawi.com/journals/wcmc/2021/8588646/>.

ABDALLAH, A. B. **Multicore Systems On-Chip: Practical Software/Hardware Design**. [S.l.]: Springer, 2013. v. 2.

ABID, S. B. et al. Nrtbox: A matlab simulink toolbox for noc switch performance evaluation and early architectural exploration using discrete event simulation. In: **2015 10th International Design & Test Symposium (IDT)**. [S.l.: s.n.], 2015. p. 96–99.

ABUMWAIS, M. E. A. A scalable interconnection scheme in many-core systems. **Computers, Materials & Continua**, v. 77, n. 1, p. 615–632, 2023. ISSN 1546-2226.

AFSHARPOUR, S.; PATOOGHY, A.; FAZELI, M. Performance/energy aware task migration algorithm for many-core chips. **IET Computers & Digital Techniques**, Wiley Online Library, v. 10, n. 4, p. 165–173, 2016.

AGHAEI, B. et al. Network adapter architectures in network on chip: Comprehensive literature review. Kluwer Academic Publishers, USA, v. 23, n. 1, p. 321–346, mar 2020. ISSN 1386-7857.

AHMAD, K.; SETHI, M. Review of network on chip routing algorithms. **EAI Endorsed Transactions on Context-aware Systems and Applications**, v. 7, n. 22, 2020.

AHMED, M.; BAIG, M. I. An improved non-local awareness of congestion and load balanced algorithm for the communication of on chip 2D mesh-based network. **Mehran University Research Journal of Engineering and Technology**, v. 42, n. 2, p. 54, abr. 2023.

AINSWORTH, S.; JONES, T. Many-core systems for big-data computing. In: AL-HASHIMI, B.; MERRETT, G. (Ed.). **Many-Core Computing: Hardware and Software**. United Kingdom: Institution of Engineering and Technology, 2019. p. 523–544. ISBN 9781785615825.

AL-HCHAIMI, A. A. J. et al. Review of 3d networks-on-chip simulators and plugins. In: **2021 IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia)**. [S.l.: s.n.], 2021. p. 17–20.

ALIMI, I. A. et al. Network-on-chip topologies: Potentials, technical challenges, recent advances and research direction. **Network-on-Chip-Architecture, Optimization, and Design Explorations**, IntechOpen, 2021.

ALVES, S. S.; OLIVEIRA, S. A.; NETO, A. R. R. A novel educational timetabling solution through recursive genetic algorithms. In: **2015 Latin America Congress on Computational Intelligence (LA-CCI)**. [S.l.: s.n.], 2015. p. 1–6.

AMIN, W. et al. Performance evaluation of application mapping approaches for network-on-chip designs. **IEEE Access**, v. 8, p. 63607–63631, 2020.

AN, J. et al. Fault tolerant xy-yx routing algorithm supporting backtracking strategy for noc. In: **2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)**. [S.l.: s.n.], 2021. p. 632–635.

ATAGOZIYEV, M. Networks on chip: Topology, switching, routing. In: . [s.n.], 2009. Disponível em: <https://api.semanticscholar.org/CorpusID:60188126>.

AUSAVARUNGNIRUN, R.; MUTLU, O. Energy-efficient deflection-based on-chip networks: Topology, routing, flow control. **arXiv preprint arXiv:2112.02516**, 2021.

BABAIE, S. et al. Online-structural testing of routers in network on chip. **World Applied Sciences Journal**, v. 14, p. 1374–1383, 01 2011.

BABOLI, M.; HUSIN, N. S.; MARSONO, M. N. A comprehensive evaluation of direct and indirect network-on-chip topologies. In: **Proceedings of the 2014 international conference on industrial engineering and operations management**. [S.l.: s.n.], 2014. p. 2081–2090.

BALAKRISHNAN, M. T.; VENKATESH, T.; BHASKAR, A. V. Design and implementation of congestion aware router for network-on-chip. **Integration**, Elsevier, v. 88, p. 43–57, 2023.

BALKIND, J. et al. Openpiton: An open source manycore research framework. **Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems**, 2016.

BENCHEHIDA, C. et al. **Mapping Hard Real-time Tasks on Network-on-Chip Manycore Architectures**. [S.l.: s.n.], 2021.

BENINI, L.; MICHELI, G. Networks on chips: A new soc paradigm. **Computer**, v. 35, p. 70–78, 02 2002.

BERTSIMAS, D.; TSITSIKLIS, J. Simulated Annealing. **Statistical Science**, Institute of Mathematical Statistics, v. 8, n. 1, p. 10 – 15, 1993.

BHANU, P. V. et al. Torus topology based fault-tolerant network-on-chip design with flexible spare core placement. In: **2018 14th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)**. [S.l.: s.n.], 2018. p. 97–100.

BHASKAR, A. V.; VENKATESH, T. Performance analysis of network-on-chip in many-core processors. **Journal of Parallel and Distributed Computing**, v. 147, p. 196–208, 2021.

BHASKAR, V. A study of network-on-chip performance. In: **Proceedings of the 2021 Thirteenth International Conference on Contemporary Computing**. New York, NY, USA: Association for Computing Machinery, 2021. (IC3-2021), p. 37–42. ISBN 9781450389204.

BHATT, N.; CHAUHAN, N. R. Genetic algorithm applications on job shop scheduling problem: A review. In: **2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI)**. [S.l.: s.n.], 2015. p. 7–14.

BHOWMIK, B. et al. Locating open-channels in octagon networks on chip-microprocessors. In: **2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)**. [S.l.: s.n.], 2020. p. 200–205.

BINKERT, N. et al. The gem5 simulator. **ACM SIGARCH Computer Architecture News**, ACM, v. 39, n. 2, p. 1–7, May 2011.

BJERREGAARD, T.; MAHADEVAN, S. A survey of research and practices of network-on-chip. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 38, n. 1, p. 1–es, 2006.

BONNEY, C. et al. Fault tolerant task mapping on many-core arrays. In: IEEE. **2016 IEEE Symposium Series on Computational Intelligence (SSCI)**. [S.l.], 2016. p. 1–8.

BOUKEDJAR, A.; LALAMI, M. E.; EL-BAZ, D. Parallel branch and bound on a cpu-gpu system. In: **2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing**. [S.l.: s.n.], 2012. p. 392–398.

BROWN, J. W. Adaptive network on chip routing using the turn model. In: . [s.n.], 2013. Disponível em: <https://api.semanticscholar.org/CorpusID:111026217>.

CARDOSO, J.; COUTINHO, J.; DINIZ, P. **Embedded Computing for High Performance: Efficient Mapping of Computations Using Customization, Code Transformations and Compilation**. [S.l.: s.n.], 2017. 1-297 p.

CARDOSO, J. M.; COUTINHO, J. G. F.; DINIZ, P. C. Chapter 8 - additional topics. In: CARDOSO, J. M.; COUTINHO, J. G. F.; DINIZ, P. C. (Ed.). **Embedded Computing for High Performance**. Boston: Morgan Kaufmann, 2017. p. 255–280. ISBN 978-0-12-804189-5.

CATANIA, V. et al. Noxim: An open, extensible and cycle-accurate network on chip simulator. In: **2015 IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP)**. [S.l.: s.n.], 2015. p. 162–163.

CHANG, W.; YUBAI, L.; SONG, C. Design and simulation of a torus topology for network on chip* *this project was supported by the national natural science fundation of china (60575031). **Journal of Systems Engineering and Electronics**, v. 19, n. 4, p. 694–701, 2008. ISSN 1004-4132.

CHEMLI, B.; ZITOUNI, A. Design of a network on chip router based on turn model. In: **2015 16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)**. [S.l.: s.n.], 2015. p. 85–88.

CHEN, J.; LI, C.; GILLARD, P. Network-on-chip (noc) topologies and performance: a review. In: **Proceedings of the 2011 Newfoundland Electrical and Computer Engineering Conference (NECEC)**. [S.l.: s.n.], 2011. p. 1–6.

COTA, É.; AMORY, A.; LUBASZEWSKI, M. **Reliability, Availability and Serviceability of Networks-on-Chip**. [S.l.: s.n.], 2012. ISBN 978-1-4614-0790-4.

DALLY, W.; TOWLES, B. Route packets, not wires: on-chip interconnection networks. In: **Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232)**. [S.l.: s.n.], 2001. p. 684–689.

DALLY, W.; TOWLES, B. **Principles and Practices of Interconnection Networks**. [S.l.]: Elsevier Science, 2004. (The Morgan Kaufmann Series in Computer Architecture and Design). ISBN 9780122007514.

DALZOTTO, A. E. et al. Dynamic mapping for many-cores using management application organization. In: **2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)**. [S.l.: s.n.], 2021. p. 1–6.

DHARWAD, S. Design and implementation of minimal adaptive west first algorithm for noc router architecture. In: . [S.l.: s.n.], 2013.

DICK, R. **TGFF: Task Graphs For Free**. 2022. <https://robertdick.org/projects/tgff/>. Acesso em: dezembro de 2022.

DICK, R.; RHODES, D.; WOLF, W. Tgff: task graphs for free. In: **Proceedings of the Sixth International Workshop on Hardware/Software Codesign. (CODES/CASHE'98)**. [S.l.: s.n.], 1998. p. 97–101.

DOMAN, D. **Engineering the CMOS Library: Enhancing Digital Design Kits for Competitive Silicon**. [S.l.]: Wiley, 2012. ISBN 9781118273135.

DORIGO, M.; CARO, G. D. Ant colony optimization: a new meta-heuristic. In: **Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)**. [S.l.: s.n.], 1999. v. 2, p. 1470–1477 Vol. 2.

DUMITRASCU, F. et al. Flexible mpsoc platform with fast interconnect exploration for optimal system performance for a specific application. **2008 Design, Automation and Test in Europe**, v. 2, p. 32, 01 2006.

EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. In: **MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science**. [S.l.: s.n.], 1995. p. 39–43.

FANG, J.; YU, T.; WEI, Z. Improved ant colony algorithm based on task scale in network on chip (noc) mapping. **Electronics**, v. 9, n. 1, 2020. ISSN 2079-9292.

FENG, C. The algorithm based on network routing on chip. **Applied and Computational Engineering**, v. 12, p. 206–213, 09 2023.

GARLAND, M.; KIRK, D. B. Understanding throughput-oriented architectures. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 53, n. 11, p. 58–66, nov 2010. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/1839676.1839694>.

GAWISH, E. K.; EL-KHARASHI, M. W.; ABU-ELYAZEED, M. Variability-tolerant routing algorithms for networks-on-chip. **Microprocessors and Microsystems**, Elsevier, v. 38, n. 8, p. 1037–1045, 2014.

GROSS, J. L.; YELLEN, J.; ANDERSON, M. **Graph theory and its applications**. [S.l.]: Chapman and Hall/CRC, 2018.

GU, H. A review of research on network-on-chip simulator. In: MA, M. (Ed.). **Communication Systems and Information Technology**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 103–110. ISBN 978-3-642-21762-3.

GULZARI, U. et al. A low latency and low power indirect topology for on-chip communication. **PLOS ONE**, v. 14, p. e0222759, 10 2019.

HALLIDAY, D.; RESNICK, R.; WALKER, J. **Fundamentos da Física - Mecânica - Volume 1**. [S.l.]: LTC, 2023. ISBN 9788521637226.

HAO, P. et al. Comparison of 2d mesh routing algorithm in noc. In: IEEE. **2011 9th IEEE International Conference on ASIC**. [S.l.], 2011. p. 791–795.

HASSAN, A. S.; MORGAN, A. A.; EL-KHARASHI, M. W. Clustered networks-on-chip: Simulationandperformanceevaluation. **International Journal of Computing and Digital Systems**, University of Bahrain, v. 6, n. 02, p. 51–61, 2017.

HENNESSY, J. L.; PATTERSON, D. A. **Computer Architecture: A Quantitative Approach**. 5. ed. [S.l.]: Morgan Kaufmann, 2012. ISBN 978-0-12-383872-8.

HENTGES, G.; ABDELREHIM, M. Odd-even flexible router for high performance network-on-chips. In: **2021 International Conference on Computational Science and Computational Intelligence (CSCI)**. [S.l.: s.n.], 2021. p. 1817–1820.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence**. [S.l.]: MIT Press, 1992. (Bradford book). ISBN 9780585038445.

HU, J.; MARCULESCU, R. Energy-aware mapping for tile-based noc architectures under performance constraints. In: **Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)**. [S.l.: s.n.], 2003. p. 233–239.

HU, J.; MARCULESCU, R. Energy-and performance-aware mapping for regular noc architectures. **IEEE Transactions on computer-aided design of integrated circuits and systems**, IEEE, v. 24, n. 4, p. 551–562, 2005.

ISSARIYAKUL, T.; HOSSAIN, E. **Introduction to Network Simulator NS2**. 2nd. ed. [S.l.]: Springer Publishing Company, Incorporated, 2011. ISBN 1461414059.

JAIN, L. Nirgam: A simulator for noc interconnect routing and applications modeling. In: **Design, Automation and Test in Europe Conference and Exhibition (DATE)**. [S.l.: s.n.], 2007. p. 1–2.

JAIN, R. **The Art of Computer Systems Performance Analysis: Techniques For Experimental Design, Measurement, Simulation, and Modeling, NY: Wiley**. [S.l.: s.n.], 1991. ISBN 0471503361.

JEANG, Y.-L.; JOU, J.; HUANG, W.-H. A binary tree based methodology for designing an application specific network-on-chip (asnoc). **IEICE Transactions**, v. 88-A, p. 3531–3538, 12 2005.

JEANG, Y.-L. et al. An adaptive routing algorithm for mesh-tree architecture in network-on-chip designs. In: **2008 3rd International Conference on Innovative Computing Information and Control**. [S.l.: s.n.], 2008. p. 182–182.

JIANG, N. et al. A detailed and flexible cycle-accurate network-on-chip simulator. In: **2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)**. [S.l.: s.n.], 2013. p. 86–96.

KAHNG, A. et al. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In: **Proceedings of the Conference on Design, Automation and Test in Europe**. [S.l.: s.n.], 2009. p. 423–428.

KAHNG, A. B.; LIN, B.; NATH, S. Orion3.0: A comprehensive noc router estimation tool. **IEEE Embedded Systems Letters**, v. 7, n. 2, p. 41–45, 2015.

KALITA, A. et al. A topology for network-on-chip. In: **2016 International Conference on Information Communication and Embedded Systems (ICICES)**. [S.l.: s.n.], 2016. p. 1–7.

KAMAL, R.; GOYAL, P.; NEHRA, V. Network on chip: Topologies, routing, implementation. v. 4, 02 2012.

KAMATH, A.; SAXENA, G.; TALAWAR, B. Analysis of ring topology for noc architecture. In: **2015 International Conference on Computing and Network Communications (CoCoNet)**. [S.l.: s.n.], 2015. p. 381–388.

KANSAKAR, P.; MUNIR, A. A design space exploration methodology for parameter optimization in multicore processors. **IEEE Transactions on Parallel and Distributed Systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 29, n. 1, p. 2–15, jan. 2018. ISSN 1045-9219.

KARIM, F.; NGUYEN, A.; DEY, S. An interconnect architecture for networking systems on chips. **IEEE micro**, IEEE, v. 22, n. 5, p. 36–45, 2002.

KARP, R. M. An introduction to randomized algorithms. **Discrete Applied Mathematics**, Elsevier, v. 34, n. 1-3, p. 165–201, 1991.

KATOCH, S.; CHAUHAN, S. S.; KUMAR, V. A review on genetic algorithm: past, present, and future. **Multimedia tools and applications**, Springer, v. 80, p. 8091–8126, 2021.

KAZEMPOUR, V.; FEDOROVA, A.; ALAGHEBAND, P. Performance implications of cache affinity on multicore processors. In: SPRINGER. **Euro-Par 2008–Parallel Processing: 14th International Euro-Par Conference, Las Palmas de Gran Canaria, Spain, August 26-29, 2008. Proceedings 14**. [S.l.], 2008. p. 151–161.

KHAN, M. A.; ANSARI, A. Q. An efficient tree-based topology for network-on-chip. In: **2011 World Congress on Information and Communication Technologies**. [S.l.: s.n.], 2011. p. 1316–1321.

KHAN, S. et al. Comparative analysis of network-on-chip simulation tools. **IET Computers & Digital Techniques**, v. 12, 09 2017.

KHAN, S. et al. Comparative analysis of network-on-chip simulation tools. **IET Computers & Digital Techniques**, v. 12, 09 2017.

KREUTZ, M. et al. Design space exploration comparing homogeneous and heterogeneous network-on-chip architectures. In: **2005 18th Symposium on Integrated Circuits and Systems Design**. [S.l.: s.n.], 2005. p. 190–195.

KUMAR, A.; TALAWAR, B. Machine learning based framework to predict performance evaluation of on-chip networks. In: **2018 Eleventh International Conference on Contemporary Computing (IC3)**. [S.l.: s.n.], 2018. p. 1–6.

KUMAR, A. S.; RAO, T. H. An adaptive core mapping algorithm on noc for future heterogeneous system-on-chip. **Computers and Electrical Engineering**, Elsevier, v. 95, p. 107441, 2021.

KUMAR, S. et al. A network on chip architecture and design methodology. In: IEEE. **Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002**. [S.l.], 2002. p. 117–124.

LI, M. et al. Potential based routing in b-p2i- architecture. In: IEEE. **The International Conference on Information Networking 2013 (ICOIN)**. [S.l.], 2013. p. 71–76.

LIU, J.; ZHENG, L.-R.; TENHUNEN, H. A guaranteed-throughput switch for network-on-chip. In: **Proceedings. 2003 International Symposium on System-on-Chip (IEEE Cat. No.03EX748)**. [S.l.: s.n.], 2003. p. 31–34.

LIU, Y. Design of hybrid wireless network on chip and maximum three hops routing algorithm. In: **2021 3rd International Conference on Applied Machine Learning (ICAML)**. [S.l.: s.n.], 2021. p. 211–216.

LIU, Y.; KATO, S.; EDAHIRO, M. Analysis of memory system of tiled many-core processors. **IEEE Access**, v. 7, p. 18964–18977, 2019.

LIU, Y. et al. Automatic endometrial segmentation in ultrasound images using deep learning. In: IEEE. **2022 IEEE 15th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)**. [S.l.], 2022. p. 67–71.

LU, Z.; SANDER, I.; JANTSCH, A. Refinement of a perfectly synchronous communication model onto nostrum noc best-effort communication service. In: . [s.n.], 2005. Disponível em: <https://api.semanticscholar.org/CorpusID:4847512>.

LU, Z. et al. Nnse: Nostrum network-on-chip simulation environment. In: **Proceedings of the SSoCC**. [S.l.: s.n.], 2002. p. 1.

MAHENDRA, C.; GAIKWAD, M.; PATRIKAR, R. Review of xy routing algorithm for network-on-chip architecture. **International Journal of Computer and Communication Technology**, 10 2016.

MANDAL, S. et al. Analytical performance modeling of nocs under priority arbitration and bursty traffic. 07 2020.

MATHEW, A.; AMUDHA, P.; SIVAKUMARI, S. Deep learning techniques: An overview. In: HASSANIEN, A. E.; BHATNAGAR, R.; DARWISH, A. (Ed.). **Advanced Machine Learning Technologies and Applications**. Singapore: Springer Singapore, 2021. p. 599–608. ISBN 978-981-15-3383-9.

MATLAB. **version 7.10.0 (R2010a)**. Natick, Massachusetts: The MathWorks Inc., 2010.

MATOUSSI, O. Noc performance model for efficient network latency estimation. In: IEEE. **2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)**. [S.l.], 2021. p. 994–999.

MATTSON, T.; WIJNGAART, R.; FRUMKIN, M. Programming the intel 80-core network-on-a-chip terascale processor. In: . [S.l.: s.n.], 2008. p. 38.

MEHMOOD, F. et al. An efficient and cost effective application mapping for network-on-chip using andean condor algorithm. **Journal of Network and Computer Applications**, Elsevier, v. 200, p. 103319, 2022.

MESSAOUDI, K. et al. Network-on-chip application mapping using genetic algorithm for a complex hardware implementation of video encoders. In: **2020 1st International Conference on Communications, Control Systems and Signal Processing (CCSSP)**. [S.l.: s.n.], 2020. p. 175–179.

MILLBERG, M. et al. Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip. In: **Proceedings Design, Automation and Test in Europe Conference and Exhibition**. [S.l.: s.n.], 2004. v. 2, p. 890–895 Vol.2.

MILLER, J. E. et al. Graphite: A distributed parallel simulator for multicores. In: **The 16th IEEE International Symposium on High-Performance Computer Architecture (HPCA)**. [S.l.: s.n.], 2010.

MINZHENG, J.; YUNZHONG, Z.; FANGFA, F. Fault-tolerant routing method of noc system based on clustering. In: **2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)**. [S.l.: s.n.], 2016. p. 543–547.

MISHRA, P.; CHARLES, S. **Network-on-chip security and privacy**. [S.l.]: Springer, 2021.

MONDAL, B. Artificial intelligence: State of the art. In: BALAS, V. E.; KUMAR, R.; SRIVASTAVA, R. (Ed.). **Recent Trends and Advances in Artificial Intelligence and Internet of Things**. Cham: Springer International Publishing, 2020. p. 389–425. ISBN 978-3-030-32644-9.

MOORE, G. Cramming more components onto integrated circuits. **Proceedings of the IEEE**, v. 86, n. 1, p. 82–85, 1998.

MUHSEN, Y. R. et al. Enhancing noc-based mpsoc performance: A predictive approach with ann and guaranteed convergence arithmetic optimization algorithm. **IEEE Access**, v. 11, p. 90143–90157, 2023.

Naresh Kumar Reddy, B.; KUMAR, A. S. Evaluating the effectiveness of bat optimization in an adaptive and energy-efficient network-on-chip routing framework. **Journal of Parallel and Distributed Computing**, v. 188, p. 104853, 2024. ISSN 0743-7315.

NOROLLAH, A. et al. Pat-noxim: A precise power & thermal cycle-accurate noc simulator. In: **2018 31st IEEE International System-on-Chip Conference (SOCC)**. [S.l.: s.n.], 2018. p. 163–168.

OGRAS, U.; MARCULESCU, R. Prediction-based flow control for network-on-chip traffic. In: **2006 43rd ACM/IEEE Design Automation Conference**. [S.l.: s.n.], 2006. p. 839–844.

OLIVEIRA, S. d. S.; CARVALHO, B. M. d.; KREUTZ, M. E. Network-on-chip irregular topology optimization for real-time and non-real-time applications. **Micromachines**, v. 12, n. 10, 2021. ISSN 2072-666X.

PAN, P.-Q.; PAN, P.-Q. Integer linear programming (ilp). **Linear Programming Computation**, Springer, p. 275–294, 2014.

PARKHANI, S. A.; TEHRE, V. A. A survey of different topologies for network-on-chip architecture. In: . [s.n.], 2016. Disponível em: <https://api.semanticscholar.org/CorpusID:2694094>.

PEDRINO, E. C.; LIMA, D. P. de; TEMPESTI, G. A multiobjective metaheuristic approach for morphological filters on many-core architectures. **Integrated Computer-Aided Engineering**, IOS Press, v. 26, n. 4, p. 383–397, 2019.

PHING, N. Y. et al. Performance analysis of the impact of design parameters to network-on-chip (noc) architecture. In: SPRINGER. **Recent Trends in Information and Communication Technology: Proceedings of the 2nd International Conference of Reliable Information and Communication Technology (IRICT 2017)**. [S.l.], 2018. p. 237–246.

PHING, N. Y. et al. Performances analysis of reducing router in ring and mesh topology for network-on-chip (noc) architecture. **Indonesian Journal of Electrical Engineering and Computer Science**, v. 14, n. 2, p. 802–809, 2019.

PHING, n. yen et al. Topology design of extended torus and ring for low latency network-on-chip architecture. **TELKOMNIKA (Telecommunication Computing Electronics and Control)**, v. 15, p. 869, 03 2017.

PINHEIRO, R.; ROMA, N.; TOMÁS, P. A cross-core performance model for heterogeneous many-core architectures. In: DUTRA, I. et al. (Ed.). **High Performance Computing for Computational Science – VECPAR 2016**. Cham: Springer International Publishing, 2017. p. 101–111. ISBN 978-3-319-61982-8.

POURMOHSENI, B. et al. Hybrid application mapping for composable many-core systems: Overview and future perspective. **Journal of Low Power Electronics and Applications**, v. 10, n. 4, 2020. ISSN 2079-9268.

RANE, U. V.; GAD, R. S.; PANEM, C. Design of network on chip (noc) computing node for mesh topology using soft-core nios-ii processor. **Journal of Physics: Conference Series**, IOP Publishing, v. 1921, n. 1, p. 012075, may 2021. Disponível em: <https://dx.doi.org/10.1088/1742-6596/1921/1/012075>.

RANTALA, V. et al. **Network on chip routing algorithms**. [S.l.]: Citeseer, 2006.

REDDY, B. N. K.; KAR, S. Performance evaluation of modified mesh-based noc architecture. **Computers and Electrical Engineering**, v. 104, p. 108404, 2022. ISSN 0045-7906.

RILEY, G. F.; HENDERSON, T. R. The ns-3 network simulator. In: WEHRLE, K.; GÜNEŞ, M.; GROSS, J. (Ed.). **Modeling and Tools for Network Simulation**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

p. 15–34. ISBN 978-3-642-12331-3. Disponível em: <https://doi.org/10.1007/978-3-642-12331-3_2>.

SADEGHI, M.; NEZHAD, A. R.; ZAVAREH, F. M. A new suggestion for improvement of mesh topology on noc. In: **2016 5th International Conference on Computer Science and Network Technology (ICCSNT)**. [S.l.: s.n.], 2016. p. 667–671.

SALEEM, S. et al. A survey on dynamic application mapping approaches for real-time network-on-chip-based platforms. **IEEE Access**, v. 11, p. 122694–122721, 2023.

SAMBANGI, R.; MANGHNANI, H.; CHATTOPADHYAY, S. Lpnet: A dnn based latency prediction technique for application mapping in network-on-chip design. **Microprocessors and Microsystems**, Elsevier, v. 87, p. 104370, 2021.

SAMBANGI, R. et al. Application mapping onto manycore processor architectures using active search framework. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, IEEE Computer Society, Los Alamitos, CA, USA, v. 31, n. 06, p. 789–801, jun 2023. ISSN 1557-9999.

SANTHOSH, G. Design of network on chip with an arbiter. In: . [s.n.], 2015. Disponível em: <https://api.semanticscholar.org/CorpusID:264613243>.

SEILER, L. et al. Larrabee: A many-core x86 architecture for visual computing. **IEEE Micro**, v. 29, n. 1, p. 10–21, 2009.

SHARMA, A. et al. Metaheuristics-based routing optimization in on-chip network. In: **Proceedings of the 2023 Fifteenth International Conference on Contemporary Computing, IC3-2023**. Noida, India: Association for Computing Machinery, 2023. p. 18–23. ISBN 9798400700224.

SHARMA, A. et al. Metaheuristics-based routing optimization in on-chip network. In: **Proceedings of the 2023 Fifteenth International Conference on Contemporary Computing**. New York, NY, USA: Association for Computing Machinery, 2023. (IC3-2023), p. 18–23. ISBN 9798400700224.

SIKANDAR, S. et al. An optimized nature-inspired metaheuristic algorithm for application mapping in 2d-noc. **Sensors**, v. 21, n. 15, p. 5102, 2021.

SIKANDAR, S. et al. An optimized nature-inspired metaheuristic algorithm for application mapping in 2d-noc. **Sensors**, MDPI, v. 21, n. 15, p. 5102, 2021.

SINGH, A. et al. Mapping on multi/many-core systems: Survey of current and emerging trends. In: ASSOCIATION FOR COMPUTING MACHINERY. **Proceedings of the 50th Annual Design Automation Conference**. [S.l.], 2013. p. 1–10.

SINGH, A. K. et al. A survey and comparative study of hard and soft real-time dynamic resource allocation strategies for multi-/many-core systems. **ACM Computing Surveys (CSUR)**, Association for Computing Machinery, v. 50, n. 2, 2017.

SINGH, A. K. et al. **Design Space Exploration and Resource Management of Multi/Many-Core Systems**. [S.l.]: MDPI - Multidisciplinary Digital Publishing Institute, 2021.

SOMISETTY, R. et al. Congestion aware negative first routing with fair arbitration for network on chip. In: **2022 6th International Conference on Computing Methodologies and Communication (ICCMC)**. [S.l.: s.n.], 2022. p. 215–220.

SONG, Y.; LIN, B. Uniform minimal first: Latency reduction in throughput-optimal oblivious routing for mesh-based networks-on-chip. **IEEE Embedded Systems Letters**, v. 11, n. 3, p. 81–84, 2019.

SONG, Z.; MA, G.; DALEI, S. Hierarchical star: An optimal noc topology for high-performance soc design. **Computer and Computational Sciences, International Multi-Symposiums on**, v. 0, p. 158–163, 10 2008.

STILLMAKER, A. et al. Scalable energy-efficient parallel sorting on a fine-grained many-core processor array. **Journal of Parallel and Distributed Computing**, v. 138, p. 32–47, 2020. ISSN 0743-7315.

SUBOH, S. et al. Analytical modeling and evaluation of network-on-chip architectures. In: IEEE. **2010 International Conference on High Performance Computing & Simulation**. [S.l.], 2010. p. 615–622.

TAJARY, A.; MORSHEDLOU, H. A simulated annealing-based throughput-aware task mapping algorithm for manycore processors. **J AI Data Min**, Shahrood University of Technology, v. 10, n. 3, p. 311–320, 2022.

TATAS, K. et al. **Designing 2D and 3D network-on-chip architectures**. [S.l.]: Springer, 2014.

THID, R.; MILLBERG, M.; JANTSCH, A. Evaluating noc communication backbones with simulation. In: IEEE CONFERENCE PROCEEDINGS. **The**

IEEE NorChip Conference, Riga, Latvia, Nov 10-11, 2003. [S.l.], 2003. p. 27–30.

TIAN, Y. et al. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization. **IEEE Computational Intelligence Magazine**, v. 12, n. 4, p. 73–87, 2017.

TINETTI, F. G. Computer architecture: A quantitative approach. **Journal of Computer Science & Technology**, Graduate Network of Argentine Universities with Computer Science Schools . . . , v. 8, n. 3, p. 168–170, 2008.

TRAN, A. T.; BAAS, B. M. Noctweak: a highly parameterizable simulator for early exploration of performance and energy of networks on-chip. In: . [S.l.: s.n.], 2012.

TRIK, M. et al. A hybrid selection strategy based on traffic analysis for improving performance in networks on chip. **Journal of Sensors**, Hindawi, v. 2022, 2022.

TSAI, W.-C. et al. Networks on chips: structure and design methodologies. **Journal of Electrical and Computer Engineering**, Hindawi Limited London, UK, United Kingdom, v. 2012, p. 2–2, 2012.

UMA, R.; SAROJADEVI, H.; SANJU, V. Network-on-chip (noc)-routing techniques: A study and analysis. In: IEEE. **2019 Global Conference for Advancement in Technology (GCAT)**. [S.l.], 2019. p. 1–6.

UMAPATHY, S.; SHAH, M.; WANG, N. Encircle routing: An efficient deterministic network on chip routing algorithm. In: IEEE. **2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)**. [S.l.], 2018. p. 895–899.

VILLAESCUSA, D. G.; RIVAS, M. A.; HARBOUR, M. G. Response-time analysis of mesh-based many-core systems. **Journal of Systems Architecture**, v. 134, p. 102762, 2023. ISSN 1383-7621.

VIPIN, K.; KALOMIROS, J. Asyncbtree: Revisiting binary tree topology for efficient fpga-based noc implementation. **International Journal of Reconfigurable Computing**, Hindawi, v. 2019, p. 7239858, 2019. ISSN 1687-7195.

WANG, H. et al. Orion: A power-performance simulator for interconnection networks. In: **Proceedings of the 35th Annual ACM/IEEE International Symposium on Microarchitecture**. [S.l.: s.n.], 2002. p. 294–305.

WEICHSLGARTNER, A. et al. **Invasive Computing for mapping parallel programs to many-core architectures**. [S.l.]: Springer, 2018. v. 1.

XU, Y.; TANIGUCHI, I.; TOMIYAMA, H. Static mapping of parallelizable tasks under deadline constraints. **IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences**, The Institute of Electronics, Information and Communication Engineers, v. 100, n. 7, p. 1500–1502, 2017.

YADAV, S. et al. Dynamic fault injection model for on-chip 2d mesh network. In: **Proceedings of the 7th ACM India Computing Conference**. New York, NY, USA: Association for Computing Machinery, 2014. (COMPUTE '14). ISBN 9781605588148.

YAN, X.; MENG, J.; CHEN, Z. Design of processors. In: WANG, Y. et al. (Ed.). **Handbook of Integrated Circuit Industry**. Singapore: Springer Nature Singapore, 2024. p. 757–777. ISBN 978-981-99-2836-1.

YE, T.; BENINI, L.; MICHELI, G. Analysis of power consumption on switch fabrics in network routers. 08 2002.

YU, J. et al. Tpnoc: An efficient topology reconfigurable noc generator. In: **Proceedings of the Great Lakes Symposium on VLSI 2023**. New York, NY, USA: Association for Computing Machinery, 2023. (GLSVLSI '23), p. 77–82. ISBN 9798400701252.

YUE, K. et al. A greedy approach to tolerate defect cores for multimedia applications. In: **2011 9th IEEE Symposium on Embedded Systems for Real-Time Multimedia**. [S.l.: s.n.], 2011. p. 112–119.

ZARRIN, J.; AGUIAR, R.; BARRACA, J. Manycore simulation for peta-scale system design: Motivation, tools, challenges and prospects. **Simulation Modelling Practice and Theory**, v. 72, p. 168–201, 2017. ISSN 1569-190X.

ZHANG, W. et al. Comparison research between xy and odd-even routing algorithm of a 2-dimension 3x3 mesh topology network-on-chip. In: **2009 WRI Global Congress on Intelligent Systems**. [S.l.: s.n.], 2009. v. 3, p. 329–333.

ZHANG, W. et al. Comparison research between xy and odd-even routing algorithm of a 2-dimension 3x3 mesh topology network-on-chip. In: **2009 WRI Global Congress on Intelligent Systems**. [S.l.: s.n.], 2009. v. 3, p. 329–333.

ZHANG, X. et al. A survey of machine learning for network-on-chips. **Journal of Parallel and Distributed Computing**, v. 186, p. 104778, 2024. ISSN 0743-7315.

ZHU, H.; PANDE, P.; GRECU, C. Performance evaluation of adaptive routing algorithms for achieving fault tolerance in noc fabrics. In: . [S.l.: s.n.], 2007. p. 42 – 47. ISBN 978-1-4244-1027-9.

Apêndice A

Tabelas Complementares - Experimento 2

Este apêndice detalha 23 combinações de parâmetros de mapeamento do Experimento 2, voltadas para a otimização do consumo de energia em arquiteturas *many-core* com NoCs. As tabelas incluídas complementam as 4 combinações discutidas no corpo do texto, fornecendo uma base ampla para análises adicionais sobre diferentes estratégias de mapeamento. Este material suporta uma investigação mais profunda dos métodos de mapeamento e seu impacto na eficiência energética, ampliando o escopo de pesquisa além do discutido no documento principal.

Tabela A.1 – Desempenho dos Algoritmos de Mapeamento (conf. 0): GA (troca, 0.02, 5), ACO (0.1, 1, 1), PSO (0.4, 1.5, 1.5), TS ((nº de tarefas), 10, 5), e SA (Tajary et al., 2022) (100, 0.01, 0.85).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
$Grafo_9$	**Média**	354.00	581.00	424.00	370.00	350.00
	Desvio Padrão	9.00	75.00	106.00	21.00	0.00
	Melhor	**350.00**	402.00	**350.00**	**350.00**	**350.00**
$Grafo_{16}$	**Média**	558.00	1188.00	1109.00	692.00	488.00
	Desvio Padrão	39.00	155.00	110.00	52.00	8.00
	Melhor	486.00	819.00	832.00	583.00	**473.00**
$Grafo_{25}$	**Média**	1119.00	2636.00	2426.00	1528.00	1029.00
	Desvio Padrão	76.00	232.00	221.00	138.00	31.00
	Melhor	959.00	2088.00	2001.00	1198.00	*948.00*
$Grafo_{36}$	**Média**	2026.00	4782.00	4636.00	2953.00	1928.00
	Desvio Padrão	110.00	376.00	376.00	236.00	49.00
	Melhor	**1759.00**	3989.00	3951.00	2468.00	1801.00
$Grafo_{49}$	**Média**	2611.00	6168.00	6116.00	3800.00	2454.00
	Desvio Padrão	137.00	521.00	400.00	291.00	63.00
	Melhor	2342.00	5100.00	5218.00	3114.00	**2303.00**
$Grafo_{64}$	**Média**	4671.00	10528.00	10143.00	6597.00	4091.00
	Desvio Padrão	223.00	579.00	656.00	457.00	106.00
	Melhor	4202.00	9369.00	8825.00	5652.00	**3716.00**
$Grafo_{81}$	**Média**	6217.00	13704.00	13186.00	8582.00	5222.00
	Desvio Padrão	364.00	689.00	832.00	598.00	156.00
	Melhor	5579.00	12188.00	11197.00	7329.00	**4677.00**
$Grafo_{100}$	**Média**	7863.00	17264.00	16703.00	10910.00	6577.00
	Desvio Padrão	573.00	932.00	1154.00	686.00	213.00
	Melhor	6678.00	14592.00	13349.00	8802.00	**5778.00**

Tabela A.2 – Desempenho dos Algoritmos de Mapeamento (conf. 1): GA (inversão, 0.02, 50), ACO (0.1, 1, 5), PSO (0.4, 1.5, 2.0), TS ((nº de tarefas), 10, 10), e SA (Tajary et al., 2022) (100, 0.01, 0.90).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
$Grafo_9$	Média	354.00	581.00	424.00	370.00	350.00
	Desvio Padrão	9.00	75.00	106.00	21.00	0.00
	Melhor	**350.00**	402.00	**350.00**	**350.00**	**350.00**
$Grafo_{16}$	Média	558.00	1188.00	1109.00	692.00	488.00
	Desvio Padrão	39.00	155.00	110.00	52.00	8.00
	Melhor	486.00	819.00	832.00	583.00	**473.00**
$Grafo_{25}$	Média	1119.00	2636.00	2426.00	1528.00	1029.00
	Desvio Padrão	76.00	232.00	221.00	138.00	31.00
	Melhor	959.00	2088.00	2001.00	1198.00	**948.00**
$Grafo_{36}$	Média	2026.00	4782.00	4636.00	2953.00	1928.00
	Desvio Padrão	110.00	376.00	376.00	236.00	49.00
	Melhor	**1759.00**	3989.00	3951.00	2468.00	1801.00
$Grafo_{49}$	Média	2611.00	6168.00	6116.00	3800.00	2454.00
	Desvio Padrão	137.00	521.00	400.00	291.00	63.00
	Melhor	2342.00	5100.00	5218.00	3114.00	**2303.00**
$Grafo_{64}$	Média	4671.00	10528.00	10143.00	6597.00	4091.00
	Desvio Padrão	223.00	579.00	656.00	457.00	106.00
	Melhor	4202.00	9369.00	8825.00	5652.00	**3716.00**
$Grafo_{81}$	Média	6217.00	13704.00	13186.00	8582.00	5222.00
	Desvio Padrão	364.00	825.00	595.00	622.00	142.00
	Melhor	5571.00	11940.00	11709.00	7242.00	**4804.00**
$Grafo_{100}$	Média	10364.00	21025.00	20010.00	13474.00	8239.00
	Desvio Padrão	386.00	880.00	825.00	870.00	191.00
	Melhor	9377.00	19258.00	18213.00	11666.00	**7623.00**

Tabela A.3 – Desempenho dos Algoritmos de Mapeamento (conf. 2): GA (troca, 0.02, 20), ACO (0.1, 1, 10), PSO (0.4, 1.5, 2.5), TS (nº de tarefas, 10, 15), e SA (Tajary et al., 2022) (100, 0.01, 0.95).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
Grafo$_{9}$	**Média**	354.00	559.00	446.00	357.00	350.00
	Desvio Padrão	9.00	78.00	105.00	10.00	0.00
	Melhor	**350.00**	383.00	**350.00**	**350.00**	**350.00**
Grafo$_{16}$	**Média**	566.00	1164.00	1144.00	589.00	488.00
	Desvio Padrão	37.00	129.00	164.00	48.00	8.00
	Melhor	484.00	886.00	801.00	488.00	**473.00**
Grafo$_{25}$	**Média**	1095.00	2624.00	2470.00	1219.00	1023.00
	Desvio Padrão	65.00	239.00	248.00	91.00	27.00
	Melhor	927.00	1935.00	1926.00	1018.00	*897.00*
Grafo$_{36}$	**Média**	2024.00	4833.00	4736.00	2266.00	1928.00
	Desvio Padrão	117.00	360.00	352.00	173.00	45.00
	Melhor	1825.00	4057.00	3854.00	1797.00	**1796.00**
Grafo$_{49}$	**Média**	2598.00	6219.00	6084.00	2902.00	2434.00
	Desvio Padrão	139.00	444.00	421.00	181.00	55.00
	Melhor	**2237.00**	5114.00	5095.00	2580.00	2299.00
Grafo$_{64}$	**Média**	4637.00	10337.00	10183.00	4854.00	4092.00
	Desvio Padrão	229.00	544.00	629.00	364.00	98.00
	Melhor	4044.00	8870.00	8690.00	4160.00	**3697.00**
Grafo$_{81}$	**Média**	6253.00	13726.00	13253.00	6290.00	5195.00
	Desvio Padrão	292.00	721.00	831.00	437.00	118.00
	Melhor	5724.00	12350.00	11697.00	5226.00	**4913.00**
Grafo$_{100}$	**Média**	10283.00	20763.00	20248.00	9804.00	8176.00
	Desvio Padrão	416.00	977.00	1258.00	627.00	192.00
	Melhor	9171.00	18582.00	18223.00	8418.00	**7681.00**

Tabela A.4 – Desempenho dos Algoritmos de Mapeamento (confi. 3): GA (troca, 0.1, 5), ACO (0.1, 2, 1), PSO (0.4, 2.0, 1.5), TS (nº de tarefas, 20, 5), e SA (Tajary et al., 2022) (100, 0.02, 0.85).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
Grafo$_{9}$	**Média**	355.00	559.00	414.00	360.00	350.00
	Desvio Padrão	9.00	78.00	77.00	14.00	0.00
	Melhor	**350.00**	383.00	**350.00**	**350.00**	**350.00**
Grafo$_{16}$	**Média**	555.00	1164.00	1113.00	662.00	486.00
	Desvio Padrão	34.00	129.00	139.00	49.00	7.00
	Melhor	**473.00**	886.00	776.00	545.00	**473.00**
Grafo$_{25}$	**Média**	1052.00	2624.00	2486.00	1482.00	1017.00
	Desvio Padrão	72.00	239.00	258.00	107.00	26.00
	Melhor	**876.00**	1935.00	1868.00	1264.00	946.00
Grafo$_{36}$	**Média**	1841.00	4833.00	4613.00	2803.00	1913.00
	Desvio Padrão	108.00	360.00	374.00	208.00	43.00
	Melhor	**1627.00**	4057.00	3668.00	2287.00	1795.00
Grafo$_{49}$	**Média**	2159.00	6219.00	6024.00	3646.00	2433.00
	Desvio Padrão	113.00	444.00	428.00	201.00	61.00
	Melhor	**1880.00**	5114.00	4931.00	3067.00	2220.00
Grafo$_{64}$	**Média**	3676.00	10337.00	10064.00	6206.00	4070.00
	Desvio Padrão	197.00	544.00	592.00	371.00	97.00
	Melhor	**3329.00**	8870.00	8553.00	5529.00	3769.00
Grafo$_{81}$	**Média**	4793.00	13726.00	13226.00	8016.00	5166.00
	Desvio Padrão	229.00	721.00	852.00	503.00	133.00
	Melhor	**4289.00**	12350.00	10916.00	6989.00	4724.00
Grafo$_{100}$	**Média**	7931.00	20763.00	20288.00	12815.00	8157.00
	Desvio Padrão	301.00	977.00	1007.00	890.00	161.00
	Melhor	**7344.00**	18582.00	18379.00	11049.00	7714.00

Tabela A.5 – Desempenho dos Algoritmos de Mapeamento (conf. 4): GA (troca, 0.1, 50), ACO (0.1, 2, 5), PSO (0.4, 2.0, 2.0), TS (nº de tarefas, 20, 10), e SA (Tajary et al., 2022) (100, 0.02, 0.90).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
Grafo$_{9}$	**Média**	354.00	576.00	509.00	355.00	350.00
	Desvio Padrão	8.00	72.00	106.00	10.00	0.00
	Melhor	**350.00**	402.00	**350.00**	**350.00**	**350.00**
Grafo$_{16}$	**Média**	556.00	1183.00	1178.00	585.00	485.00
	Desvio Padrão	37.00	151.00	126.00	42.00	7.00
	Melhor	480.00	838.00	889.00	**484.00**	**473.00**
Grafo$_{25}$	**Média**	1059.00	2640.00	2559.00	1293.00	1014.00
	Desvio Padrão	77.00	246.00	192.00	87.00	27.00
	Melhor	**876.00**	2088.00	2225.00	1073.00	900.00
Grafo$_{36}$	**Média**	1839.00	4719.00	4613.00	2383.00	1909.00
	Desvio Padrão	95.00	396.00	336.00	157.00	49.00
	Melhor	1503.00	3735.00	3814.00	2067.00	**1735.00**
Grafo$_{49}$	**Média**	2167.00	6150.00	6131.00	3040.00	2427.00
	Desvio Padrão	109.00	327.00	431.00	184.00	61.00
	Melhor	1880.00	5380.00	4943.00	2605.00	**2184.00**
Grafo$_{64}$	**Média**	3652.00	10467.00	10119.00	5172.00	4043.00
	Desvio Padrão	187.00	606.00	681.00	338.00	96.00
	Melhor	3298.00	9089.00	8768.00	4272.00	**3735.00**
Grafo$_{81}$	**Média**	4799.00	13579.00	13330.00	6604.00	5149.00
	Desvio Padrão	221.00	658.00	697.00	456.00	112.00
	Melhor	4112.00	12207.00	11325.00	5714.00	**4821.00**
Grafo$_{100}$	**Média**	7893.00	21034.00	20173.00	10350.00	8132.00
	Desvio Padrão	313.00	1012.00	1068.00	627.00	153.00
	Melhor	7152.00	18571.00	17661.00	9045.00	**7691.00**

Tabela A.6 – Desempenho dos Algoritmos de Mapeamento (conf. 5): GA (troca, 0.1, 20), ACO (0.1, 2, 10), PSO (0.4, 2.0, 2.5), TS (nº de tarefas, 20, 15), e SA (Tajary et al., 2022) (100, 0.02, 0.95).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
Grafo$_{9}$	**Média**	354.00	579.00	397.00	354.00	350.00
	Desvio Padrão	8.00	72.00	88.00	8.00	0.00
	Melhor	**350.00**	402.00	**350.00**	**350.00**	**350.00**
Grafo$_{16}$	**Média**	555.00	1197.00	1026.00	560.00	485.00
	Desvio Padrão	37.00	145.00	178.00	34.00	7.00
	Melhor	480.00	838.00	568.00	492.00	**473.00**
Grafo$_{25}$	**Média**	1066.00	2629.00	2384.00	1175.00	1011.00
	Desvio Padrão	70.00	237.00	204.00	81.00	25.00
	Melhor	**876.00**	2088.00	1891.00	992.00	940.00
Grafo$_{36}$	**Média**	1829.00	4705.00	4596.00	2162.00	1894.00
	Desvio Padrão	95.00	419.00	371.00	127.00	50.00
	Melhor	**1562.00**	3735.00	3898.00	1874.00	1746.00
Grafo$_{49}$	**Média**	2187.00	6105.00	5988.00	2727.00	2414.00
	Desvio Padrão	118.00	488.00	446.00	161.00	54.00
	Melhor	**1880.00**	5100.00	4787.00	2336.00	2195.00
Grafo$_{64}$	**Média**	5568.00	10484.00	10042.00	4537.00	4054.00
	Desvio Padrão	247.00	629.00	704.00	266.00	102.00
	Melhor	5103.00	9284.00	8798.00	3845.00	**3744.00**
Grafo$_{81}$	**Média**	4761.00	13694.00	13218.00	5827.00	5143.00
	Desvio Padrão	247.00	697.00	841.00	426.00	117.00
	Melhor	**4799.00**	13694.00	13218.00	6604.00	5143.00
Grafo$_{100}$	**Média**	7926.00	20807.00	20309.00	10350.00	8086.00
	Desvio Padrão	285.00	1072.00	1165.00	587.00	178.00
	Melhor	7893.00	20807.00	20309.00	9290.00	**7625.00**

Tabela A.7 – Desempenho dos Algoritmos de Mapeamento (conf.7): GA (troca, 0.05, 50), ACO (0.1, 3, 5), PSO (0.4, 2.5, 2.0), TS (nº de tarefas, 30, 10), e SA (Tajary et al., 2022) (100, 0.03, 0.90).

Grafos de Aplicação	Métricas	Adaptado do PlatEMO			Implementado	Implementado
		GA	ACO	PSO	TS	SA (Tajary et. al. 2022)
$Grafo_9$	**Média**	357.00	578.00	550.00	353.00	350.00
	Desvio Padrão	11.00	72.00	100.00	7.00	0.00
	Melhor	**350.00**	402.00	**350.00**	**350.00**	**350.00**
$Grafo_{16}$	**Média**	552.00	1223.00	1191.00	574.00	482.00
	Desvio Padrão	40.00	136.00	147.00	34.00	5.00
	Melhor	**482.00**	936.00	752.00	486.00	**473.00**
$Grafo_{25}$	**Média**	1073.00	2557.00	2479.00	1231.00	997.00
	Desvio Padrão	73.00	257.00	229.00	78.00	24.00
	Melhor	**910.00**	2082.00	2012.00	1067.00	**927.00**
$Grafo_{36}$	**Média**	1858.00	4539.00	4685.00	2299.00	1881.00
	Desvio Padrão	101.00	339.00	331.00	123.00	45.00
	Melhor	1634.00	3735.00	4012.00	2021.00	**1718.00**
$Grafo_{49}$	**Média**	2322.00	6187.00	6110.00	2961.00	2388.00
	Desvio Padrão	121.00	409.00	462.00	179.00	55.00
	Melhor	2084.00	5380.00	4867.00	2646.00	**2224.00**
$Grafo_{64}$	**Média**	6008.00	10424.00	10240.00	4924.00	3997.00
	Desvio Padrão	273.00	689.00	689.00	288.00	90.00
	Melhor	5318.00	8528.00	8726.00	4316.00	**3684.00**
$Grafo_{81}$	**Média**	5395.00	13657.00	13478.00	6367.00	5113.00
	Desvio Padrão	235.00	686.00	695.00	335.00	96.00
	Melhor	4729.00	12104.00	12123.00	5651.00	**4800.00**
$Grafo_{100}$	**Média**	8892.00	21236.00	20322.00	10125.00	8000.00
	Desvio Padrão	320.00	992.00	1030.00	544.00	175.00
	Melhor	8196.00	19417.00	17987.00	9040.00	**7388.00**

Tabela A.8 – Desempenho dos Algoritmos de Mapeamento (conf. 8): GA (troca, 0.05, 20), ACO (0.1, 3, 10), PSO (0.4, 2.5, 2.5), TS (nº de tarefas, 30, 15), e SA (Tajary et al., 2022) (100, 0.03, 0.95).

Grafos de Aplicação	Métricas	Adaptado do PlatEMO			Implementado	Implementado
		GA	ACO	PSO	TS	SA (Tajary et. al. 2022)
$Grafo_9$	**Média**	353.00	571.00	496.00	353.00	350.00
	Desvio Padrão	7.00	79.00	87.00	7.00	0.00
	Melhor	**350.00**	409.00	**350.00**	**350.00**	**350.00**
$Grafo_{16}$	**Média**	554.00	1221.00	1193.00	574.00	479.00
	Desvio Padrão	39.00	145.00	156.00	34.00	5.00
	Melhor	487.00	914.00	858.00	486.00	**473.00**
$Grafo_{25}$	**Média**	1048.00	2591.00	2435.00	1231.00	992.00
	Desvio Padrão	66.00	237.00	258.00	78.00	22.00
	Melhor	**906.00**	2111.00	1899.00	1067.00	923.00
$Grafo_{36}$	**Média**	1867.00	4749.00	4661.00	2299.00	1863.00
	Desvio Padrão	98.00	363.00	335.00	123.00	41.00
	Melhor	**1683.00**	3840.00	3997.00	2021.00	1738.00
$Grafo_{49}$	**Média**	2294.00	6224.00	6174.00	2961.00	2362.00
	Desvio Padrão	104.00	342.00	416.00	179.00	55.00
	Melhor	**2084.00**	5531.00	5115.00	2646.00	2147.00
$Grafo_{64}$	**Média**	5970.00	10572.00	10105.00	4924.00	3968.00
	Desvio Padrão	255.00	676.00	620.00	288.00	91.00
	Melhor	5418.00	9074.00	8482.00	4316.00	**3673.00**
$Grafo_{81}$	**Média**	5371.00	13584.00	13264.00	6367.00	5056.00
	Desvio Padrão	259.00	685.00	748.00	335.00	94.00
	Melhor	**4691.00**	11988.00	11251.00	5651.00	4779.00
$Grafo_{100}$	**Média**	8840.00	21050.00	20429.00	10125.00	7963.00
	Desvio Padrão	369.00	966.00	1101.00	544.00	145.00
	Melhor	8128.00	18756.00	17588.00	9040.00	**7550.00**

Tabela A.9 – Desempenho dos Algoritmos de Mapeamento (conf.10): GA (inserção, 0.02, 50), ACO (0.5, 1, 5), PSO (0.6, 1.5, 2.0), TS (nº de tarefas*1.5, 10, 10), e SA (Tajary et al., 2022) (150, 0.01, 0.90).

Grafos de Aplicação	Métricas	Adaptado do PlatEMO			Implementado	Implementado
		GA	ACO	PSO	TS	SA (Tajary et. al. 2022)
$Grafo_9$	Média	370.00	573.00	534.00	360.00	350.00
	Desvio Padrão	18.00	73.00	82.00	14.00	0.00
	Melhor	**350.00**	402.00	372.00	**350.00**	**350.00**
$Grafo_{16}$	Média	650.00	1163.00	1165.00	614.00	489.00
	Desvio Padrão	61.00	160.00	126.00	44.00	8.00
	Melhor	**350.00**	819.00	951.00	511.00	473.00
$Grafo_{25}$	Média	1410.00	2601.00	2455.00	1335.00	1031.00
	Desvio Padrão	121.00	228.00	214.00	106.00	28.00
	Melhor	1165.00	2174.00	2039.00	1088.00	**948.00**
$Grafo_{36}$	Média	2707.00	4684.00	4724.00	2499.00	1929.00
	Desvio Padrão	222.00	386.00	435.00	201.00	45.00
	Melhor	2249.00	3735.00	3975.00	2091.00	**1760.00**
$Grafo_{49}$	Média	3750.00	6252.00	6086.00	3180.00	2441.00
	Desvio Padrão	291.00	485.00	450.00	204.00	56.00
	Melhor	3008.00	5335.00	5322.00	2677.00	**2271.00**
$Grafo_{64}$	Média	6546.00	10528.00	10105.00	5334.00	4102.00
	Desvio Padrão	425.00	522.00	567.00	379.00	86.00
	Melhor	5863.00	8917.00	9003.00	4697.00	**3823.00**
$Grafo_{81}$	Média	8930.00	13801.00	13222.00	6968.00	5185.00
	Desvio Padrão	516.00	691.00	791.00	508.00	139.00
	Melhor	7723.00	12093.00	11551.00	5960.00	**4680.00**
$Grafo_{100}$	Média	14388.00	21050.00	20594.00	10941.00	8207.00
	Desvio Padrão	782.00	1157.00	1091.00	813.00	189.00
	Melhor	12440.00	18566.00	18207.00	9445.00	**7611.00**

Tabela A.10 – Desempenho dos Algoritmos de Mapeamento (conf. 11): GA (inserção, 0.02, 20), ACO (0.5, 1, 10), PSO (0.6, 1.5, 2.5), TS (nº de tarefas*1.5, 10, 15), e SA (Tajary et al., 2022) (150, 0.01, 0.95).

Grafos de Aplicação	Métricas	Adaptado do PlatEMO			Implementado	Implementado
		GA	ACO	PSO	TS	SA (Tajary et. al. 2022)
$Grafo_9$	Média	371.00	572.00	549.00	357.00	350.00
	Desvio Padrão	18.00	82.00	83.00	11.00	0.00
	Melhor	**350.00**	390.00	**350.00**	**350.00**	**350.00**
$Grafo_{16}$	Média	653.00	1143.00	1176.00	577.00	488.00
	Desvio Padrão	57.00	149.00	139.00	40.00	7.00
	Melhor	526.00	829.00	877.00	501.00	**473.00**
$Grafo_{25}$	Média	1411.00	2524.00	2465.00	1235.00	1022.00
	Desvio Padrão	146.00	233.00	235.00	92.00	24.00
	Melhor	1157.00	1912.00	1912.00	1031.00	**961.00**
$Grafo_{36}$	Média	2759.00	4707.00	4707.00	2280.00	1911.00
	Desvio Padrão	229.00	340.00	345.00	162.00	52.00
	Melhor	2183.00	3920.00	4154.00	1966.00	**1787.00**
$Grafo_{49}$	Média	3748.00	6233.00	6095.00	2873.00	2437.00
	Desvio Padrão	289.00	382.00	372.00	194.00	73.00
	Melhor	2895.00	5159.00	5415.00	2516.00	**2209.00**
$Grafo_{64}$	Média	6625.00	10418.00	10045.00	4892.00	4063.00
	Desvio Padrão	395.00	584.00	731.00	323.00	98.00
	Melhor	5863.00	9074.00	8651.00	4098.00	**3774.00**
$Grafo_{81}$	Média	8877.00	13650.00	13285.00	6164.00	5184.00
	Desvio Padrão	492.00	676.00	721.00	472.00	114.00
	Melhor	7789.00	11908.00	5181.00	5803.00	**4877.00**
$Grafo_{100}$	Média	14422.00	21269.00	20316.00	9835.00	8153.00
	Desvio Padrão	713.00	1010.00	1114.00	680.00	216.00
	Melhor	12874.00	18675.00	17365.00	8524.00	**7488.00**

Tabela A.11 – Desempenho dos Algoritmos de Mapeamento (conf. 13): GA (inserção, 0.1, 50), ACO (0.5, 2, 5), PSO (0.6, 2.0, 2.0), TS (nº de tarefas*1.5, 20, 10), e SA (Tajary et al., 2022) (150, 0.02, 0.90).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
Grafo$_{9}$	**Média**	364.00	579.00	536.00	353.00	350.00
	Desvio Padrão	18.00	72.00	76.00	8.00	0.00
	Melhor	**350.00**	402.00	383.00	**350.00**	**350.00**
Grafo$_{16}$	**Média**	647.00	1198.00	1181.00	586.00	485.00
	Desvio Padrão	52.00	143.00	131.00	37.00	6.00
	Melhor	553.00	838.00	830.00	502.00	**473**
Grafo$_{25}$	**Média**	1366.00	2607.00	2560.00	1267.00	1014.00
	Desvio Padrão	104.00	242.00	224.00	80.00	29.00
	Melhor	1138.00	2032.00	2111.00	1096.00	**913.00**
Grafo$_{36}$	**Média**	2524.00	4651.00	4636.00	2357.00	1901.00
	Desvio Padrão	240.00	352.00	292.00	154.00	45.00
	Melhor	2068.00	3735.00	3968.00	2058.00	**1769.00**
Grafo$_{49}$	**Média**	3352.00	6269.00	6138.00	3003.00	2421.00
	Desvio Padrão	301.00	412.00	367.00	185.00	59.00
	Melhor	2839.00	5380.00	5338.00	2635.00	**2254.00**
Grafo$_{64}$	**Média**	5777.00	10299.00	10036.00	5112.00	4053.00
	Desvio Padrão	443.00	554.00	612.00	351.00	99.00
	Melhor	4350.00	9369.00	8317.00	4344.00	**3722.00**
Grafo$_{81}$	**Média**	7925.00	13705.00	13300.00	6541.00	5172.00
	Desvio Padrão	363.00	767.00	867.00	382.00	125.00
	Melhor	7062.00	11931.00	11718.00	5625.00	**4715.00**
Grafo$_{100}$	**Média**	12537.00	21166.00	20609.00	10311.00	8126.00
	Desvio Padrão	605.00	1005.00	944.00	611.00	175.00
	Melhor	11393.00	19205.00	18381.00	8723.00	**7630.00**

Tabela A.12 – Desempenho dos Algoritmos de Mapeamento (conf. 14): GA (inserção, 0.1, 50), ACO (0.5, 2, 10), PSO (0.6, 2.0, 2.5), TS (nº de tarefas*1.5, 20, 15), e SA (Tajary et al., 2022) (150, 0.02, 0.95).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
Grafo$_{9}$	**Média**	364.00	578.00	585.00	354.00	350.00
	Desvio Padrão	18.00	71.00	79.00	8.00	0.00
	Melhor	**350.00**	402.00	427.00	**350.00**	**350.00**
Grafo$_{16}$	**Média**	649.00	1200.00	1183.00	554.00	484.00
	Desvio Padrão	61.00	137.00	133.00	34.00	5.00
	Melhor	**515.00**	936.00	913.00	**486.00**	**743.00**
Grafo$_{25}$	**Média**	1337.00	2609.00	2518.00	1166.00	1012.00
	Desvio Padrão	115.00	279.00	229.00	69.00	29.00
	Melhor	1128.00	2082.00	2040.00	1014.00	**910.00**
Grafo$_{36}$	**Média**	2524.00	4731.00	4660.00	2167.00	1911.00
	Desvio Padrão	192.00	430.00	384.00	138.00	44.00
	Melhor	2126.00	3735.00	4052.00	1902.00	**1801.00**
Grafo$_{49}$	**Média**	3304.00	6179.00	6072.00	2712.00	2401.00
	Desvio Padrão	281.00	410.00	391.00	179.00	61.00
	Melhor	2860.00	5406.00	5351.00	2396.00	**2209.00**
Grafo$_{64}$	**Média**	5793.00	10462.00	10029.00	4547.00	4035.00
	Desvio Padrão	417.00	535.00	528.00	255.00	99.00
	Melhor	4806.00	9369.00	9052.00	3971.00	**3685.00**
Grafo$_{81}$	**Média**	7713.00	13685.00	13324.00	5817.00	5130.00
	Desvio Padrão	531.00	789.00	800.00	375.00	118.00
	Melhor	6677.00	12093.00	11615.00	5056.00	**4778.00**
Grafo$_{100}$	**Média**	12750.00	20881.00	20285.00	9281.00	8096.00
	Desvio Padrão	567.00	988.00	889.00	534.00	154.00
	Melhor	11704.00	18966.00	18680.00	8142.00	**7624.00**

Tabela A.13 – Desempenho dos Algoritmos de Mapeamento (conf. 15): GA (inserção, 0.05, 5), ACO (0.5, 3, 1), PSO (0.6, 2.5, 1.5), TS (nº de tarefas*1.5, 30, 5), e SA (Tajary et al., 2022) (150, 0.03, 0.85).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
Grafo$_{9}$	**Média**	365.00	575.00	552.00	356.00	350.00
	Desvio Padrão	15.00	72.00	65.00	10.00	0.00
	Melhor	**350.00**	402.00	384.00	**350.00**	**350.00**
Grafo$_{16}$	**Média**	641.00	1200.00	1160.00	649.00	483.00
	Desvio Padrão	54.00	154.00	138.00	37.00	6.00
	Melhor	510.00	819.00	841.00	559.00	**473.00**
Grafo$_{25}$	**Média**	1397.00	2578.00	2480.00	1435.00	1007.00
	Desvio Padrão	124.00	253.00	231.00	93.00	26.00
	Melhor	1171.00	1937.00	1986.00	1219.00	**935.00**
Grafo$_{36}$	**Média**	2627.00	4700.00	4640.00	2714.00	1891.00
	Desvio Padrão	207.00	348.00	425.00	170.00	48.00
	Melhor	2048.00	3735.00	3656.00	2234.00	**1693.00**
Grafo$_{49}$	**Média**	3467.00	6136.00	5970.00	3555.00	2408.00
	Desvio Padrão	281.00	472.00	489.00	185.00	49.00
	Melhor	2882.00	5100.00	5017.00	2997.00	**2236.00**
Grafo$_{64}$	**Média**	6006.00	10516.00	10296.00	6057.00	4030.00
	Desvio Padrão	402.00	641.00	648.00	332.00	91.00
	Melhor	4505.00	8905.00	9056.00	5378.00	**3668.00**
Grafo$_{81}$	**Média**	8061.00	13656.00	13231.00	7875.00	5137.00
	Desvio Padrão	494.00	705.00	654.00	463.00	92.00
	Melhor	6853.00	12003.00	12185.00	6889.00	**4915.00**
Grafo$_{100}$	**Média**	13479.00	20995.00	20448.00	12341.00	8022.00
	Desvio Padrão	637.00	965.00	1036.00	663.00	160.00
	Melhor	12155.00	18773.00	17876.00	11057.00	**7621.00**

Tabela A.14 – Desempenho dos Algoritmos de Mapeamento (conf. 16): GA (inserção, 0.05, 50), ACO (0.5, 3, 5), PSO (0.6, 2.5, 2.0), TS (nº de tarefas*1.5, 30, 5), e SA (Tajary et al., 2022) (150, 0.03, 0.90).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
Grafo$_{9}$	**Média**	366.00	581.00	578.00	353.00	350.00
	Desvio Padrão	16.00	75.00	79.00	8.00	0.00
	Melhor	**350.00**	402.00	428.00	**350.00**	**350.00**
Grafo$_{16}$	**Média**	661.00	1178.00	1170.00	578.00	481.00
	Desvio Padrão	63.00	143.00	120.00	34.00	5.00
	Melhor	551.00	838.00	908.00	480.00	473.00
Grafo$_{25}$	**Média**	1394.00	2596.00	2555.00	1234.00	998.00
	Desvio Padrão	130.00	256.00	242.00	78.00	26.00
	Melhor	1137.00	2032.00	2067.00	1016.00	**912.00**
Grafo$_{36}$	**Média**	2655.00	4706.00	4714.00	2307.00	1879.00
	Desvio Padrão	253.00	338.00	325.00	157.00	48.00
	Melhor	2116.00	4072.00	4129.00	2002.00	**1715.00**
Grafo$_{49}$	**Média**	3453.00	6213.00	6118.00	2942.00	2385.00
	Desvio Padrão	266.00	431.00	456.00	159.00	49.00
	Melhor	2944.00	5216.00	5156.00	2565.00	**2246.00**
Grafo$_{64}$	**Média**	6148.00	10266.00	10015.00	5004.00	4009.00
	Desvio Padrão	424.00	581.00	613.00	272.00	78.00
	Melhor	5400.00	9284.00	8678.00	4361.00	**3817.00**
Grafo$_{81}$	**Média**	8267.00	13588.00	13238.00	6345.00	5081.00
	Desvio Padrão	464.00	920.00	844.00	403.00	116.00
	Melhor	7089.00	11826.00	10638.00	5515.00	**4686.00**
Grafo$_{100}$	**Média**	13372.00	20936.00	20248.00	9999.00	8038.00
	Desvio Padrão	594.00	1001.00	993.00	480.00	170.00
	Melhor	12022.00	19052.00	18522.00	8732.00	**7537.00**

Tabela A.15 – Desempenho dos Algoritmos de Mapeamento (conf. 17): GA (inserção, 0.05, 20), ACO (0.5, 3, 10), PSO (0.6, 2.5, 2.5), TS (nº de tarefas*1.5, 30, 15), e SA (Tajary et al., 2022) (150, 0.03, 0.95).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
Grafo$_{9}$	**Média**	365.00	595.00	539.00	353.00	350.00
	Desvio Padrão	15.00	78.00	102.00	7.00	0.00
	Melhor	**350.00**	**450.00**	351.00	**350.00**	**350.00**
Grafo$_{16}$	**Média**	639.00	1178.00	1137.00	545.00	480.00
	Desvio Padrão	54.00	136.00	126.00	28.00	5.00
	Melhor	516.00	858.00	852.00	482.00	**473.00**
Grafo$_{25}$	**Média**	1402.00	2565.00	2494.00	1132.00	985.00
	Desvio Padrão	132.00	218.00	189.00	63.00	28.00
	Melhor	1094.00	2052.00	2054.00	981.00	**904.00**
Grafo$_{36}$	**Média**	2631.00	4750.00	4676.00	2123.00	1858.00
	Desvio Padrão	209.00	387.00	323.00	128.00	49.00
	Melhor	2193.00	3985.00	3998.00	1876.00	**1735.00**
Grafo$_{49}$	**Média**	3401.00	6223.00	6029.00	2658.00	2376.00
	Desvio Padrão	243.00	454.00	387.00	136.00	49.00
	Melhor	2882.00	5165.00	5052.00	2368.00	**2240.00**
Grafo$_{64}$	**Média**	6023.00	10240.00	10119.00	4451.00	3959.00
	Desvio Padrão	420.00	598.00	641.00	249.00	87.00
	Melhor	4505.00	8855.00	8659.00	3859.00	**3709.00**
Grafo$_{81}$	**Média**	8066.00	13648.00	13238.00	5662.00	5048.00
	Desvio Padrão	447.00	819.00	844.00	319.00	79.00
	Melhor	7128.00	11838.00	12058.00	4961.00	**4831.00**
Grafo$_{100}$	**Média**	13376.00	20980.00	20548.00	8964.00	7970.00
	Desvio Padrão	766.00	973.00	1173.00	467.00	133.00
	Melhor	11780.00	19240.00	17919.00	8022.00	**7604.00**

Tabela A.16 – Desempenho dos Algoritmos de Mapeamento (conf. 18): GA (inversão, 0.02, 5), ACO (0.9, 1, 1), PSO (0.8, 1.5, 2.0), TS (nº de tarefas*2, 10, 5), e SA (Tajary et al., 2022) (200, 0.01, 0.85).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
Grafo$_{9}$	**Média**	365.00	575.00	574.00	374.00	350.00
	Desvio Padrão	15.00	73.00	80.00	21.00	0.00
	Melhor	**350.00**	**402.00**	411.00	**350.00**	**350.00**
Grafo$_{16}$	**Média**	641.00	1150.00	1173.00	697.00	490.00
	Desvio Padrão	60.00	132.00	139.00	62.00	9.00
	Melhor	498.00	892.00	932.00	562.00	**473.00**
Grafo$_{25}$	**Média**	1416.00	2603.00	2484.00	1561.00	1030.00
	Desvio Padrão	121.00	296.00	193.00	126.00	28.00
	Melhor	1153.00	2123.00	2088.00	1191.00	**963.00**
Grafo$_{36}$	**Média**	2685.00	4757.00	4675.00	2970.00	1923.00
	Desvio Padrão	210.00	364.00	290.00	224.00	52.00
	Melhor	2239.00	4072.00	4137.00	2521.00	**1754.00**
Grafo$_{49}$	**Média**	3834.00	6329.00	6131.00	3827.00	2445.00
	Desvio Padrão	305.00	392.00	381.00	306.00	63.00
	Melhor	3270.00	5512.00	5430.00	3247.00	**2213.00**
Grafo$_{64}$	**Média**	6624.00	10407.00	10177.00	6525.00	4111.00
	Desvio Padrão	366.00	655.00	637.00	462.00	94.00
	Melhor	5889.00	9150.00	8940.00	5365.00	**3898.00**
Grafo$_{81}$	**Média**	9044.00	13742.00	13226.00	8534.00	5228.00
	Desvio Padrão	429.00	749.00	704.00	628.00	122.00
	Melhor	8110.00	12367.00	11710.00	7355.00	**4919.00**
Grafo$_{100}$	**Média**	14505.00	21231.00	20039.00	13203.00	8214.00
	Desvio Padrão	720.00	1037.00	1077.00	812.00	154.00
	Melhor	12721.00	19438.00	17240.00	11354.00	**7807.00**

Tabela A.17 – Desempenho dos Algoritmos de Mapeamento (conf. 19): GA (inversão, 0.02, 50), ACO (0.9, 1, 5), PSO (0.8, 1.5, 2.0), TS (nº de tarefas*2, 10, 10), e SA (Tajary et al., 2022) (200, 0.01, 0.90).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
Grafo$_{9}$	**Média**	366.00	575.00	571.00	357.00	350.00
	Desvio Padrão	15.00	73.00	70.00	12.00	0.00
	Melhor	**350.00**	**402.00**	420.00	**350.00**	**350.00**
Grafo$_{16}$	**Média**	638.00	1158.00	1179.00	616.00	488.00
	Desvio Padrão	53.00	151.00	136.00	48.00	8.00
	Melhor	531.00	819.00	874.00	495.00	**473.00**
Grafo$_{25}$	**Média**	1398.00	2622.00	2533.00	1320.00	1022.00
	Desvio Padrão	122.00	251.00	198.00	99.00	29.00
	Melhor	1183.00	2088.00	2020.00	1100.00	**942.00**
Grafo$_{36}$	**Média**	2727.00	4640.00	4651.00	2501.00	1927.00
	Desvio Padrão	197.00	328.00	354.00	176.00	49.00
	Melhor	2277.00	3735.00	3908.00	2135.00	**1799.00**
Grafo$_{49}$	**Média**	3731.00	6188.00	6088.00	3145.00	2442.00
	Desvio Padrão	261.00	415.00	403.00	195.00	55.00
	Melhor	3071.00	5219.00	5270.00	2817.00	**2256.00**
Grafo$_{64}$	**Média**	6607.00	10542.00	10089.00	5404.00	4097.00
	Desvio Padrão	304.00	602.00	640.00	339.00	103.00
	Melhor	5982.00	9284.00	8557.00	4559.00	**3790.00**
Grafo$_{81}$	**Média**	8923.00	13704.00	13259.00	6959.00	5179.00
	Desvio Padrão	442.00	730.00	795.00	429.00	111.00
	Melhor	8163.00	12269.00	11104.00	5989.00	**4640.00**
Grafo$_{100}$	**Média**	14578.00	20779.00	20029.00	11088.00	8209.00
	Desvio Padrão	800.00	964.00	1141.00	727.00	189.00
	Melhor	13048.00	18648.00	17878.00	9661.00	**7641.00**

Tabela A.18 – Desempenho dos Algoritmos de Mapeamento (conf. 20): GA (inversão, 0.1, 50), ACO (0.9, 2, 5), PSO (0.8, 2.0, 2.0), TS (nº de tarefas*2, 20, 10), e SA (Tajary et al., 2022) (200, 0.02, 0.90).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
Grafo$_{9}$	**Média**	367.00	581.00	593.00	357.00	350.00
	Desvio Padrão	19.00	75.00	72.00	11.00	0.00
	Melhor	**350.00**	402.00	457.00	**350.00**	**350.00**
Grafo$_{16}$	**Média**	631.00	1172.00	1180.00	572.00	487.00
	Desvio Padrão	52.00	145.00	158.00	36.00	7.00
	Melhor	504.00	838.00	916.00	486.00	**473.00**
Grafo$_{25}$	**Média**	1412.00	2622.00	2478.00	1231.00	1017.00
	Desvio Padrão	139.00	269.00	216.00	93.00	30.00
	Melhor	1156.00	2032.00	2012.00	1057.00	**921.00**
Grafo$_{36}$	**Média**	2752.00	4739.00	4587.00	2289.00	1917.00
	Desvio Padrão	189.00	365.00	292.00	164.00	44.00
	Melhor	2317.00	3949.00	3900.00	1952.00	**1799.00**
Grafo$_{49}$	**Média**	3722.00	6176.00	6088.00	2885.00	2430.00
	Desvio Padrão	254.00	381.00	397.00	227.00	58.00
	Melhor	3071.00	5219.00	5256.00	2350.00	**2245.00**
Grafo$_{64}$	**Média**	6661.00	10327.00	10073.00	4888.00	4093.00
	Desvio Padrão	388.00	654.00	599.00	306.00	103.00
	Melhor	5866.00	9284.00	9151.00	4166.00	**3769.00**
Grafo$_{81}$	**Média**	9036.00	13573.00	13014.00	6222.00	5196.00
	Desvio Padrão	426.00	886.00	789.00	405.00	128.00
	Melhor	8163.00	11913.00	11351.00	5229.00	**4812.00**
Grafo$_{100}$	**Média**	14611.00	21113.00	20416.00	9865.00	8183.00
	Desvio Padrão	698.00	1156.00	1074.00	703.00	169.00
	Melhor	13283.00	18452.00	18426.00	8390.00	**7784.00**

Tabela A.19 – Desempenho dos Algoritmos de Mapeamento (conf. 22): GA (inversão, 0.02, 20), ACO (0.9, 1, 10), PSO (0.8, 1.5, 2.5), TS (nº de tarefas*2, 10, 15), e SA (Tajary et al., 2022) (200, 0.01, 0.95).

Grafos de Aplicação	Métricas	Adaptado do PlatEMO			Implementado	Implementado
		GA	ACO	PSO	TS	SA (Tajary et. al. 2022)
$\mathbf{Grafo_9}$	**Média**	369.00	576.00	552.00	354.00	350.00
	Desvio Padrão	17.00	75.00	65.00	8.00	0.00
	Melhor	**350.00**	402.00	384.00	**350.00**	**350.00**
$\mathbf{Grafo_{16}}$	**Média**	625.00	1196.00	1160.00	588.00	485.00
	Desvio Padrão	37.00	141.00	138.00	37.00	7.00
	Melhor	513.00	892.00	841.00	513.00	**473.00**
$\mathbf{Grafo_{25}}$	**Média**	1310.00	2563.00	2480.00	1269.00	1014.00
	Desvio Padrão	132.00	254.00	231.00	72.00	27.00
	Melhor	1052.00	2037.00	1986.00	1070.00	**900.00**
$\mathbf{Grafo_{36}}$	**Média**	2517.00	4720.00	4640.00	2389.00	1909.00
	Desvio Padrão	182.00	343.00	425.00	160.00	49.00
	Melhor	2142.00	3737.00	3656.00	2022.00	**1735.00**
$\mathbf{Grafo_{49}}$	**Média**	3346.00	6128.00	5970.00	3004.00	2427.00
	Desvio Padrão	260.00	418.00	489.00	192.00	61.00
	Melhor	2886.00	5350.00	5017.00	2615.00	**2184.00**
$\mathbf{Grafo_{64}}$	**Média**	5971.00	10644.00	10296.00	5064.00	4043.00
	Desvio Padrão	402.00	587.00	648.00	323.00	96.00
	Melhor	5118.00	9201.00	9056.00	4120.00	**3735.00**
$\mathbf{Grafo_{81}}$	**Média**	7987.00	13542.00	13231.00	6615.00	5149.00
	Desvio Padrão	447.00	784.00	654.00	394.00	112.00
	Melhor	6809.00	11882.00	12185.00	5775.00	**4821.00**
$\mathbf{Grafo_{100}}$	**Média**	13004.00	20968.00	20448.00	10361.00	8132.00
	Desvio Padrão	622.00	884.00	1036.00	684.00	153.00
	Melhor	11511.00	19178.00	17876.00	9063.00	**7691.00**

Tabela A.20 – Desempenho dos Algoritmos de Mapeamento (conf. 23): GA (inversão, 0.1, 20), ACO (0.9, 2, 10), PSO (0.8, 2.0, 2.5), TS (nº de tarefas*2, 20, 15), e SA (Tajary et al., 2022) (200, 0.02, 0.95).

Grafos de Aplicação	Métricas	Adaptado do PlatEMO			Implementado	Implementado
		GA	ACO	PSO	TS	SA (Tajary et. al. 2022)
$\mathbf{Grafo_9}$	**Média**	360.00	575.00	573.00	355.00	350.00
	Desvio Padrão	14.00	71.00	69.00	8.00	0.00
	Melhor	**350.00**	402.00	402.00	**350.00**	**350.00**
$\mathbf{Grafo_{16}}$	**Média**	627.00	1196.00	1135.00	557.00	483.00
	Desvio Padrão	56.00	149.00	147.00	32.00	6.00
	Melhor	517.00	819.00	858.00	486.00	**473.00**
$\mathbf{Grafo_{25}}$	**Média**	1316.00	2673.00	2494.00	1157.00	1009.00
	Desvio Padrão	140.00	252.00	217.00	70.00	27.00
	Melhor	1099.00	2236.00	1987.00	998.00	**935.00**
$\mathbf{Grafo_{36}}$	**Média**	2557.00	4677.00	4622.00	2170.00	1893.00
	Desvio Padrão	233.00	337.00	323.00	132.00	49.00
	Melhor	1930.00	3735.00	3803.00	1922.00	**1698.00**
$\mathbf{Grafo_{49}}$	**Média**	3299.00	6256.00	6121.00	2734.00	2408.00
	Desvio Padrão	233.00	482.00	398.00	172.00	54.00
	Melhor	2956.00	5365.00	5230.00	2411.00	**2272.00**
$\mathbf{Grafo_{64}}$	**Média**	5932.00	10550.00	10245.00	4596.00	4046.00
	Desvio Padrão	340.00	630.00	618.00	290.00	80.00
	Melhor	5210.00	8982.00	8427.00	3858.00	**3826.00**
$\mathbf{Grafo_{81}}$	**Média**	7934.00	13865.00	13237.00	5871.00	5116.00
	Desvio Padrão	539.00	728.00	674.00	372.00	114.00
	Melhor	6620.00	12103.00	11702.00	4928.00	**4859.00**
$\mathbf{Grafo_{100}}$	**Média**	12957.00	20928.00	20689.00	9287.00	8121.00
	Desvio Padrão	756.00	1062.00	1208.00	548.00	170.00
	Melhor	10956.00	18707.00	17297.00	8118.00	**7623.00**

Tabela A.21 – Desempenho dos Algoritmos de Mapeamento (conf. 24): GA (inversão, 0.05, 5), ACO (0.9, 3, 1), PSO (0.8, 2.0, 1.5), TS (nº de tarefas*2, 30, 5), e SA (Tajary et al., 2022) (200, 0.03, 0.85).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
Grafo$_{9}$	**Média**	364.00	575.00	557.00	358.00	350.00
	Desvio Padrão	16.00	72.00	75.00	12.00	0.00
	Melhor	**350.00**	402.00	410.00	**350.00**	**350.00**
Grafo$_{16}$	**Média**	640.00	1178.00	1171.00	650.00	489.00
	Desvio Padrão	52.00	164.00	149.00	40.00	8.00
	Melhor	488.00	819.00	856.00	554.00	**473.00**
Grafo$_{25}$	**Média**	1365.00	2605.00	2503.00	1439.00	1031.00
	Desvio Padrão	119.00	216.00	222.00	93.00	28.00
	Melhor	1113.00	2174.00	1916.00	1210.00	**948.00**
Grafo$_{36}$	**Média**	2605.00	4795.00	4629.00	2689.00	1929.00
	Desvio Padrão	215.00	384.00	295.00	135.00	45.00
	Melhor	1969.00	3989.00	3924.00	2297.00	**1760.00**
Grafo$_{49}$	**Média**	3487.00	6202.00	6043.00	3533.00	2441.00
	Desvio Padrão	194.00	535.00	434.00	206.00	56.00
	Melhor	3027.00	5100.00	5226.00	3090.00	**2271.00**
Grafo$_{64}$	**Média**	6208.00	10535.00	10154.00	6034.00	4102.00
	Desvio Padrão	372.00	551.00	657.00	335.00	86.00
	Melhor	5497.00	9555.00	8839.00	5333.00	**3823.00**
Grafo$_{81}$	**Média**	8204.00	13730.00	13272.00	7855.00	5185.00
	Desvio Padrão	425.00	750.00	685.00	434.00	139.00
	Melhor	7145.00	11826.00	11962.00	6942.00	**4680.00**
Grafo$_{100}$	**Média**	13760.00	20971.00	20594.00	12301.00	8207.00
	Desvio Padrão	656.00	1127.00	1051.00	649.00	189.00
	Melhor	12447.00	18696.00	18219.00	11136.00	**7611.00**

Tabela A.22 – Desempenho dos Algoritmos de Mapeamento (conf.25): GA (inversão, 0.05, 50), ACO (0.9, 3, 5), PSO (0.8, 2.5, 2.0), TS (nº de tarefas*2, 30, 10), e SA (Tajary et al., 2022) (200, 0.03, 0.90).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**Implementado**	**Implementado**
		GA	**ACO**	**PSO**	**TS**	**SA (Tajary et. al. 2022)**
Grafo$_{9}$	**Média**	364.00	581.00	552.00	354.00	350.00
	Desvio Padrão	16.00	75.00	72.00	8.00	0.00
	Melhor	**350.00**	402.00	409.00	**350.00**	**350.00**
Grafo$_{16}$	**Média**	640.00	1178.00	1171.00	650.00	489.00
	Desvio Padrão	52.00	164.00	149.00	40.00	8.00
	Melhor	488.00	819.00	856.00	554.00	**473.00**
Grafo$_{25}$	**Média**	1365.00	2605.00	2503.00	1439.00	1031.00
	Desvio Padrão	119.00	216.00	222.00	93.00	28.00
	Melhor	1113.00	2174.00	1916.00	1210.00	**948.00**
Grafo$_{36}$	**Média**	2605.00	4795.00	4629.00	2689.00	1929.00
	Desvio Padrão	215.00	384.00	295.00	135.00	45.00
	Melhor	1969.00	3989.00	3924.00	2297.00	**1760.00**
Grafo$_{49}$	**Média**	3487.00	6202.00	6043.00	3533.00	2441.00
	Desvio Padrão	194.00	535.00	434.00	206.00	56.00
	Melhor	3027.00	5100.00	5226.00	3090.00	**2271.00**
Grafo$_{64}$	**Média**	6214.00	10405.00	10351.00	5008.00	3995.00
	Desvio Padrão	378.00	738.00	540.00	310.00	93.00
	Melhor	5588.00	9176.00	8838.00	4230.00	**3719.00**
Grafo$_{81}$	**Média**	8333.00	13722.00	13306.00	6424.00	5116.00
	Desvio Padrão	489.00	700.00	691.00	369.00	92.00
	Melhor	7266.00	11880.00	11493.00	5538.00	**4820.00**
Grafo$_{100}$	**Média**	13567.00	20981.00	20175.00	10083.00	8024.00
	Desvio Padrão	581.00	1024.00	981.00	562.00	140.00
	Melhor	12411.00	18649.00	18294.00	8662.00	**7569.00**

Tabela A.23 – Desempenho dos Algoritmos de Mapeamento (conf. 26): GA (inversão, 0.05, 20), ACO (0.9, 3, 10), PSO (0.8, 2.5, 2.5), TS (nº de tarefas*2, 30, 15), e SA (Tajary et al., 2022) (200, 0.03, 0.95).

Grafos de Aplicação	**Métricas**	**Adaptado do PlatEMO**			**TS**	**SA (Tajary et. al. 2022)**
		GA	**ACO**	**PSO**	**TS**	**SA**
Grafo$_9$	**Média**	364.00	579.00	580.00	352.00	350.00
	Desvio Padrão	16.00	72.00	77.00	6.00	0.00
	Melhor	**350.00**	402.00	428.00	**350.00**	**350.00**
Grafo$_{16}$	**Média**	631.00	1199.00	1192.00	539.00	480.00
	Desvio Padrão	53.00	144.00	115.00	30.00	5.00
	Melhor	521.00	838.00	913.00	489.00	**473.00**
Grafo$_{25}$	**Média**	1358.00	2623.00	2586.00	1139.00	994.00
	Desvio Padrão	108.00	259.00	251.00	63.00	22.00
	Melhor	1141.00	2032.00	2219.00	1018.00	**921.00**
Grafo$_{36}$	**Média**	2629.00	4623.00	4649.00	2116.00	1863.00
	Desvio Padrão	205.00	337.00	379.00	116.00	41.00
	Melhor	2233.00	3735.00	3901.00	1884.00	**1757.00**
Grafo$_{49}$	**Média**	3479.00	6173.00	6077.00	2670.00	2366.00
	Desvio Padrão	225.00	377.00	395.00	146.00	50.00
	Melhor	2902.00	5219.00	5245.00	2330.00	**2189.00**
Grafo$_{64}$	**Média**	6113.00	10476.00	9955.00	4447.00	3989.00
	Desvio Padrão	316.00	697.00	633.00	233.00	75.00
	Melhor	5390.00	8620.00	8856.00	3928.00	**3772.00**
Grafo$_{81}$	**Média**	8397.00	13619.00	13133.00	5709.00	5063.00
	Desvio Padrão	482.00	662.00	664.00	340.00	89.00
	Melhor	7362.00	12097.00	11789.00	5113.00	**4875.00**
Grafo$_{100}$	**Média**	13436.00	21064.00	20598.00	9028.00	7970.00
	Desvio Padrão	780.00	1107.00	914.00	509.00	134.00
	Melhor	12007.00	19052.00	19076.00	7898.00	**7633.00**

Apêndice B

Tabelas Complementares - Experimento 3

Este apêndice contém seis tabelas complementares do Experimento 3, que oferecem análises adicionais, se necessárias, sobre o impacto de diversas combinações de parâmetros de mutação e suas respectivas taxas de mutação na otimização de estratégias de roteamento, tendo a métrica de energia como foco principal. Elas são apresentadas abaixo, detalhando cada configuração testada:

a) **Tabela B.1**: Mutação por troca, taxa de mutação 0.02.

b) **Tabela B.2**: Mutação por troca, taxa de mutação 0.05.

c) **Tabela B.3**: Mutação por inserção, taxa de mutação 0.1.

d) **Tabela B.4**: Mutação por inserção, taxa de mutação 0.05.

e) **Tabela B.5**: Mutação por inversão, taxa de mutação 0.1.

f) **Tabela B.6**: Mutação por inversão, taxa de mutação 0.02.

Tabela B.1 – Desempenho de Algoritmos de Roteamento (conf. 1)

Grafos	Estatísticas	West-First	Negative-First	North-Last	Odd-Even	XY	XYX
$Grafo_9$	Média	89,00	94,00	103,00	98,00	98,00	103,00
	Desvio Padrão	34,00	40,00	38,00	35,00	35,00	34,00
	Melhor	19,00	**18,00**	37,00	47,00	47,00	37,00
$Grafo_{16}$	Média	322,00	303,00	318,00	313,00	313,00	317,00
	Desvio Padrão	57,00	57,00	70,00	67,00	67,00	68,00
	Melhor	206,00	**172,00**	176,00	**152,00**	**152,00**	213,00
$Grafo_{25}$	Média	760,00	747,00	736,00	736,00	736,00	766,00
	Desvio Padrão	124,00	85,00	99,00	115,00	115,00	114,00
	Melhor	439,00	604,00	505,00	517,00	517,00	**433,00**
$Grafo_{36}$	Média	1487,00	1493,00	736,00	1464,00	1464,00	1480,00
	Desvio Padrão	162,00	139,00	99,00	141,00	141,00	166,00
	Melhor	1136,00	1156,00	**505,00**	1049,00	1049,00	1073,00
$Grafo_{49}$	Média	1867,00	1888,00	1905,00	1900,00	1900,00	1875,00
	Desvio Padrão	161,00	150,00	153,00	142,00	142,00	162,00
	Melhor	1486,00	1570,00	1633,00	1523,00	1523,00	**1443,00**
$Grafo_{64}$	Média	3292,00	3322,00	3350,00	3347,00	3347,00	3329,00
	Desvio Padrão	265,00	238,00	196,00	198,00	198,00	236,00
	Melhor	2642,00	**2580,00**	2938,00	2888,00	2888,00	2846,00
$Grafo_{81}$	Média	4496,00	4543,00	4442,00	4469,00	4469,00	4542,00
	Desvio Padrão	265,00	282,00	274,00	234,00	234,00	262,00
	Melhor	4022,00	3931,00	**3841,00**	3934,00	3934,00	3984,00
$Grafo_{100}$	Média	7504,00	7514,00	7474,00	7432,00	7432,00	7394,00
	Desvio Padrão	354,00	363,00	319,00	344,00	344,00	377,00
	Melhor	6541,00	6696,00	6644,00	6843,00	6843,00	**6416,00**

Tabela B.2 – Desempenho de Algoritmos de Roteamento (conf. 2)

Grafos	Estatísticas	West-First	Negative-First	North-Last	Odd-Even	XY	XYX
$Grafo_9$	Média	89,00	114,00	103,00	98,00	103,00	103,00
	Desvio Padrão	34,00	42,00	38,00	35,00	34,00	34,00
	Melhor	**19,00**	37,00	37,00	47,00	37,00	37,00
$Grafo_{16}$	Média	319,00	322,00	318,00	313,00	316,00	317,00
	Desvio Padrão	54,00	57,00	70,00	67,00	68,00	68,00
	Melhor	206,00	181,00	176,00	**152,00**	213,00	213,00
$Grafo_{25}$	Média	782,00	785,00	736,00	736,00	767,00	766,00
	Desvio Padrão	118,00	102,00	99,00	115,00	115,00	114,00
	Melhor	439,00	553,00	505,00	517,00	**433,00**	**433,00**
$Grafo_{36}$	Média	1484,00	1633,00	1449,00	1464,00	1437,00	1480,00
	Desvio Padrão	144,00	125,00	128,00	141,00	120,00	162,00
	Melhor	1136,00	1408,00	**1036,00**	1049,00	1206,00	1073,00
$Grafo_{49}$	Média	1887,00	2281,00	1905,00	1900,00	1882,00	1866,00
	Desvio Padrão	169,00	138,00	153,00	142,00	172,00	153,00
	Melhor	1486,00	1993,00	1633,00	1523,00	**1443,00**	**1443,00**
$Grafo_{64}$	Média	3344,00	4170,00	3350,00	3353,00	3393,00	3351,00
	Desvio Padrão	234,00	242,00	196,00	188,00	188,00	234,00
	Melhor	2759,00	3369,00	2938,00	2888,00	3026,00	**2877,00**
$Grafo_{81}$	Média	4490,00	5798,00	4444,00	4427,00	4474,00	4538,00
	Desvio Padrão	235,00	297,00	278,00	306,00	289,00	249,00
	Melhor	4022,00	5247,00	3841,00	**3771,00**	**3747,00**	4074,00
$Grafo_{100}$	Média	7478,00	9645,00	7443,00	7488,00	7564,00	7443,00
	Desvio Padrão	405,00	503,00	340,00	339,00	378,00	380,00
	Melhor	6541,00	8347,00	6644,00	6949,00	6612,00	**6416,00**

Tabela B.3 – Desempenho de Algoritmos de Roteamento (conf. 3)

Grafos	Estatísticas	West-First	Negative-First	North-Last	Odd-Even	XY	XYX
$Grafo_9$	Média	108,00	114,00	115,00	102,00	110,00	110,00
	Desvio Padrão	45,00	42,00	38,00	39,00	39,00	39,00
	Melhor	37,00	37,00	**19,00**	**19,00**	47,00	47,00
$Grafo_{16}$	Média	310,00	322,00	331,00	313,00	313,00	324,00
	Desvio Padrão	78,00	57,00	56,00	72,00	65,00	67,00
	Melhor	**111,00**	181,00	196,00	185,00	184,00	184,00
$Grafo_{25}$	Média	801,00	785,00	792,00	820,00	790,00	801,00
	Desvio Padrão	111,00	102,00	106,00	104,00	113,00	110,00
	Melhor	570,00	553,00	555,00	614,00	**537,00**	544,00
$Grafo_{36}$	Média	1638,00	1633,00	1656,00	1624,00	1643,00	1668,00
	Desvio Padrão	173,00	125,00	148,00	158,00	147,00	147,00
	Melhor	1279,00	1408,00	1342,00	**1189,00**	1237,00	1237,00
$Grafo_{49}$	Média	2247,00	2281,00	2290,00	2344,00	2254,00	2258,00
	Desvio Padrão	186,00	138,00	156,00	206,00	193,00	205,00
	Melhor	1645,00	1993,00	1911,00	1816,00	**1806,00**	1691,00
$Grafo_{64}$	Média	4175,00	4170,00	4191,00	4167,00	4138,00	4170,00
	Desvio Padrão	222,00	242,00	306,00	234,00	315,00	248,00
	Melhor	3685,00	**3369,00**	3486,00	3666,00	3580,00	3706,00
$Grafo_{81}$	Média	5798,00	5798,00	5781,00	5805,00	5757,00	5764,00
	Desvio Padrão	291,00	297,00	293,00	294,00	363,00	372,00
	Melhor	5062,00	5247,00	5182,00	**4968,00**	5105,00	5034,00
$Grafo_{100}$	Média	9748,00	9645,00	9492,00	9636,00	9541,00	9630,00
	Desvio Padrão	433,00	503,00	476,00	485,00	407,00	465,00
	Melhor	8844,00	8347,00	8527,00	8361,00	8717,00	**8265,00**

Tabela B.4 – Desempenho de Algoritmos de Roteamento (conf. 5)

Grafos	Estatísticas	West-First	Negative-First	North-Last	Odd-Even	XY	XYX
$Grafo_9$	Média	108,00	114,00	115,00	102,00	110,00	110,00
	Desvio Padrão	45,00	42,00	38,00	39,00	39,00	39,00
	Melhor	37,00	37,00	**19,00**	**19,00**	47,00	47,00
$Grafo_{16}$	Média	310,00	322,00	333,00	313,00	313,00	313,00
	Desvio Padrão	78,00	57,00	57,00	72,00	65,00	65,00
	Melhor	**111,00**	181,00	196,00	185,00	184,00	184,00
$Grafo_{25}$	Média	801,00	776,00	787,00	820,00	790,00	790,00
	Desvio Padrão	111,00	101,00	103,00	104,00	113,00	113,00
	Melhor	570,00	**553,00**	555,00	614,00	537,00	537,00
$Grafo_{36}$	Média	1638,00	1623,00	1669,00	1624,00	1643,00	1643,00
	Desvio Padrão	173,00	160,00	135,00	158,00	147,00	147,00
	Melhor	1279,00	**1283,00**	1342,00	1189,00	1237,00	1237,00
$Grafo_{49}$	Média	2247,00	2273,00	2304,00	2341,00	2254,00	2261,00
	Desvio Padrão	186,00	178,00	168,00	205,00	193,00	187,00
	Melhor	1645,00	1686,00	2049,00	1816,00	**1806,00**	**1806,00**
$Grafo_{64}$	Média	4184,00	4157,00	4203,00	4179,00	4138,00	4145,00
	Desvio Padrão	232,00	243,00	283,00	232,00	315,00	310,00
	Melhor	3685,00	**3484 ,00**	3486,00	3666,00	3580,00	3595,00
$Grafo_{81}$	Média	5764,00	5777,00	5794,00	5788,00	5749,00	5773,00
	Desvio Padrão	319,00	300,00	374,00	325,00	359,00	370,00
	Melhor	**4798,00**	5243,00	5070,00	4968,00	5105,00	5034,00
$Grafo_{100}$	Média	9660,00	9653,00	9748,00	9655,00	9611,00	9679,00
	Desvio Padrão	437,00	557,00	483,00	486,00	448,00	463,00
	Melhor	8798,00	8347,00	8614,00	8788,00	8717,00	**8265,00**

Tabela B.5 – Desempenho de Algoritmos de Roteamento (conf. 6)

Grafos	Estatísticas	West-First	Negative-First	North-Last	Odd-Even	XY	XYX
Grafo$_9$	Média	103,00	101,00	105,00	101,00	108,00	108,00
	Desvio Padrão	32,00	38,00	30,00	38,00	36,00	36,00
	Melhor	29,00	37,00	47,00	**19,00**	37,00	37,00
Grafo$_{16}$	Média	301,00	320,00	337,00	326,00	312,00	311,00
	Desvio Padrão	61,00	64,00	57,00	72,00	59,00	58,00
	Melhor	181,00	193,00	212,00	199,00	193,00	**193,00**
Grafo$_{25}$	Média	761,00	790,00	783,00	805,00	748,00	760,00
	Desvio Padrão	128,00	110,00	108,00	125,00	106,00	115,00
	Melhor	505,00	575,00	473,00	570,00	**526,00**	**526,00**
Grafo$_{36}$	Média	1536,00	1534,00	1476,00	1488,00	1528,00	1520,00
	Desvio Padrão	137,00	128,00	158,00	140,00	166,00	155,00
	Melhor	1168,00	1301,00	**1176,00**	1230,00	1172,00	1220,00
Grafo$_{49}$	Média	1983,00	2030,00	2032,00	2010,00	2001,00	2024,00
	Desvio Padrão	156,00	150,00	167,00	174,00	150,00	165,00
	Melhor	1635,00	1749,00	**1598,00**	1659,00	1667,00	1718,00
Grafo$_{64}$	Média	3655,00	3623,00	3606,00	3626,00	3629,00	3625,00
	Desvio Padrão	218,00	240,00	233,00	244,00	192,00	223,00
	Melhor	3251,00	**3075,00**	3167,00	3147,00	3078,00	3222,00
Grafo$_{81}$	Média	4894,00	4926,00	5054,00	4957,00	4931,00	4926,00
	Desvio Padrão	283,00	338,00	342,00	304,00	255,00	266,00
	Melhor	4124,00	4153,00	4400,00	**4047,00**	4235,00	4327,00
Grafo$_{100}$	Média	8338,00	8205,00	8280,00	8268,00	8261,00	8252,00
	Desvio Padrão	354,00	381,00	359,00	330,00	404,00	393,00
	Melhor	7497,00	7389,00	7637,00	7646,00	**7084,00**	7226,00

Tabela B.6 – Desempenho de Algoritmos de Roteamento (conf. 7)

Grafos	Estatísticas	West-First	Negative-First	North-Last	Odd-Even	XY	XYX
Grafo$_9$	Média	103,00	101,00	105,00	101,00	108,00	108,00
	Desvio Padrão	35,00	38,00	30,00	38,00	36,00	36,00
	Melhor	**18,00**	37,00	47,00	19,00	37,00	37,00
Grafo$_{16}$	Média	312,00	318,00	338,00	101,00	312,00	311,00
	Desvio Padrão	73,00	66,00	57,00	38,00	59,00	58,00
	Melhor	165,00	193,00	212,00	**19,00**	193,00	193,00
Grafo$_{25}$	Média	771,00	786,00	795,00	805,00	748,00	769,00
	Desvio Padrão	98,00	87,00	97,00	125,00	106,00	121,00
	Melhor	543,00	640,00	637,00	570,00	**526,00**	**526,00**
Grafo$_{36}$	Média	1535,00	1529,00	1489,00	1488,00	1533,00	1536,00
	Desvio Padrão	136,00	151,00	158,00	140,00	159,00	144,00
	Melhor	1293,00	1288,00	**1176,00**	1230,00	1172,00	1175,00
Grafo$_{49}$	Média	1983,00	2019,00	2021,00	2012,00	2026,00	2016,00
	Desvio Padrão	161,00	144,00	188,00	177,00	147,00	142,00
	Melhor	1617,00	1745,00	**1470,00**	1659,00	1718,00	1667,00
Grafo$_{64}$	Média	3631,00	3670,00	3625,00	3685,00	3579,00	3624,00
	Desvio Padrão	253,00	232,00	238,00	239,00	182,00	231,00
	Melhor	3017,00	3139,00	3029,00	3147,00	**3078,00**	**3078,00**
Grafo$_{81}$	Média	4967,00	4924,00	4968,00	5029,00	4919,00	4973,00
	Desvio Padrão	258,00	349,00	328,00	299,00	275,00	264,00
	Melhor	4444,00	4153,00	3923,00	4320,00	4327,00	4301,00
Grafo$_{100}$	Média	8251,00	8310,00	8365,00	8276,00	8249,00	8295,00
	Desvio Padrão	415,00	414,00	405,00	360,00	400,00	413,00
	Melhor	7007,00	7359,00	7479,00	7623,00	**7009,00**	7388,00