UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUCAS WARLEY RODRIGUES VIANA

ALGORITMO GENÉTICO PARALELO: UMA ABORDAGEM MEMÉTICA EM ILHAS

RUSSAS

2025

LUCAS WARLEY RODRIGUES VIANA

ALGORITMO GENÉTICO PARALELO: UMA ABORDAGEM MEMÉTICA EM ILHAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Cenez Araújo de Rezende

RUSSAS

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

R614a Rodrigues, Lucas Warley Viana.
Algoritmo genético paralelo : uma abordagem memética em ilhas / Lucas Warley Viana Rodrigues. – 2026.
70 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Ciência da Computação, Russas, 2026.
Orientação: Prof. Cenez Araújo de Rezende.

1. Algoritmos Genéticos. 2. Algoritmos Meméticos. 3. Modelo em Ilhas. I. Título.
CDD 005

LUCAS WARLEY RODRIGUES VIANA

ALGORITMO GENÉTICO PARALELO: UMA ABORDAGEM MEMÉTICA EM ILHAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em: 20/01/2026

BANCA EXAMINADORA

Prof. Dr. Cenez Araújo de Rezende (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Pablo Luiz Braga Soares
Universidade Federal do Ceará (UFC)

Prof. Dr. Markos Oliveira Freitas
Universidade Federal do Ceará (UFC)

Dedico este trabalho aos meus pais, Francisca Silvana e Ricelle Warney, e à minha irmã, Lorena Vitoria. Obrigado por nunca deixarem de acreditar em mim.

AGRADECIMENTOS

À minha mãe, Francisca Silvana, que sempre me apoiou e aconselhou, e que é, e sempre será, o maior exemplo de uma mulher forte e batalhadora; ao meu pai, Ricelle Warney, pela confiança e dedicação, sempre me ajudando com o que fosse necessário e sendo, para sempre, o maior exemplo de companheirismo que poderei ter; e à minha irmã mais nova, Lorena Vitoria, por sua companhia e por conseguir tornar os dias sempre mais alegres.

Agradeço ao Prof. Dr. Cenez Araújo de Rezende pelo tempo, paciência e dedicação na realização deste trabalho.

“O maior obstáculo da vida é a expectativa, que depende do amanhã e perde o hoje.”

(Sêneca)

RESUMO

Este trabalho investiga a aplicação de Algoritmos Genéticos Paralelos de natureza memética na resolução do Problema de Roteamento de Veículos com Capacidade (CVRP), um problema clássico de otimização combinatória amplamente estudado na literatura. A abordagem proposta utiliza o modelo em ilhas, no qual múltiplas subpopulações evoluem de forma parcialmente independente, realizando trocas periódicas de indivíduos por meio de mecanismos de migração, com o objetivo de preservar a diversidade genética e mitigar a convergência prematura. O algoritmo desenvolvido incorpora operadores genéticos específicos para problemas de permutação, diferentes estratégias de seleção e um operador de busca local baseado no método 2-opt, caracterizando uma abordagem memética. A paralelização é implementada em ambiente de memória compartilhada por meio da biblioteca OpenMP, permitindo a avaliação do impacto de distintas topologias de comunicação entre ilhas, como anel e malha. Os experimentos realizados com instâncias clássicas do CVRP analisam tanto a qualidade das soluções obtidas quanto o comportamento computacional da abordagem paralela, considerando métricas como tempo de execução, speedup e eficiência, de forma complementar. Os resultados indicam que o modelo em ilhas contribui positivamente para a obtenção de soluções de maior qualidade, ao mesmo tempo em que apresenta ganhos computacionais consistentes em ambientes multicore.

Palavras-chave: algoritmos genéticos paralelos; algoritmos meméticos; modelo em ilhas; problema de roteamento de veículos com capacidade; openMP.

ABSTRACT

This work investigates the application of parallel memetic genetic algorithms to solve the Capacitated Vehicle Routing Problem (CVRP), a classical combinatorial optimization problem widely studied in the literature. The proposed approach adopts the island model, in which multiple subpopulations evolve in a partially independent manner, periodically exchanging individuals through migration mechanisms in order to preserve genetic diversity and mitigate premature convergence. The developed algorithm incorporates genetic operators tailored for permutation-based problems, different selection strategies, and a local search operator based on the 2-opt method, characterizing a memetic approach. Parallelization is implemented in a shared-memory environment using the OpenMP library, allowing the evaluation of the impact of different communication topologies between islands, such as ring and mesh. The experimental results obtained from classical CVRP benchmark instances analyze both the quality of the solutions and the computational behavior of the parallel approach, considering metrics such as execution time, speedup, and efficiency in a complementary manner. The results indicate that the island model contributes positively to achieving higher-quality solutions, while also providing consistent computational gains in multicore environments.

Keywords: parallel genetic algorithms; memetic algorithms; island model; capacitated vehicle routing problem; openMP.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE SÍMBOLOS

$>$	Maior que
$\in$	Pertence
$\subseteq$	Contido em
$\sum$	Somatório

SUMÁRIO

1 INTRODUÇÃO

No início do século 21, boa parte do avanço no desempenho dos processadores foi altamente impulsionada pelo aumento incessante do número de transistores, assim como previa a Lei de Moore. Porém, esse crescimento encontrou limites físicos significativos, como a grande dissipação de calor, o alto consumo de energia e vazamentos de correntes elétricas, tornando inviável a evolução contínua somente pelo aumento da frequência de *clock* (Sutter, 2005). Com esse cenário, os projetistas de *hardware* aproveitaram esse número progressivo de transistores disponíveis nos *chips* mais modernos e passaram a extrair mais paralelismo do *software*, o que possibilitou o ganho de desempenho mesmo diante dos limites físicos (Olukotun; Hammond, 2005)

Conforme as arquiteturas paralelas foram se desenvolvendo, foi necessário modificar os programas para que pudessem se beneficiar dessa nova situação. Como resultado, foram criados softwares capazes de executar vários fluxos de instruções ao mesmo tempo, conhecidos como *threads*. Esse método, conhecido como paralelismo em nível de *thread*, faz um melhor uso dos recursos disponíveis nos processadores modernos. Existem dois métodos principais para possibilitar o paralelismo: o *multithreading*, que permite que múltiplas *threads* sejam executadas dentro de um único núcleo, e o *Chip Multiprocessing* (CMP),sendo uma arquitetura de processadores em que vários núcleos independentes ficam dentro de um único chip de CPU, que é extensamente utilizado em processadores multicore. Vale lembrar que existem diversas maneiras pelas quais o *multithreading* pode ocorrer, incluindo o *multithreading* por fatias de tempo, que alterna entre execuções de *threads*; o *multithreading* orientado a eventos, que usa o tempo ocioso para alternar tarefas; e o *multithreading* simultâneo (SMT), também comercializado como *Hyper-Threading*. Com a introdução do paralelismo, surgiu também a capacidade de simular e solucionar problemas mais complexos (Rauber; Rünger, 2010).

O *Capacitated Vehicle Routing Problem* (CVRP) é uma variação do clássico *Vehicle Routing Problem* (VRP), introduzido por Dantzig e Ramser (1959). Nessa versão, além de determinar a melhor rota para atender a todos os clientes, é necessário considerar a restrição de capacidade dos veículos. Essa limitação exige que a carga total atribuída a cada veículo não ultrapasse um valor máximo pré-definido. O CVRP possui aplicações práticas em diversas áreas, como entrega de mercadorias, transporte coletivo, coleta de resíduos e logística urbana. Sua formulação é similar à do Problema do Caixeiro Viajante, ou *Traveling Salesman Problem* (TSP), mas com múltiplos veículos, rotas independentes e uma restrição adicional de capacidade, o que

aumenta significativamente sua complexidade (Dantzig; Ramser, 1959).

Diante da complexidade combinatória do CVRP, técnicas de otimização robustas e flexíveis tornam-se essenciais. Os Algoritmos Genéticos são técnicas de otimização inspiradas na teoria da evolução natural, conforme descrito por Russell e Norvig (2013). Eles operam sobre uma população de indivíduos, onde cada indivíduo representa uma possível solução codificada, geralmente em forma de sequência de genes. A evolução da população ocorre por meio de operadores genéticos como seleção, cruzamento (*crossover*) e mutação, buscando maximizar uma função de aptidão que avalia a qualidade de cada solução. Em cada geração, indivíduos com melhor desempenho têm maior probabilidade de serem selecionados para reprodução, promovendo a propagação de características vantajosas. Segundo os autores, os AGs se destacam pela sua capacidade de realizar buscas eficazes em espaços de solução vastos e complexos, sendo especialmente úteis em problemas onde métodos determinísticos são ineficientes ou impraticáveis. Além disso, sua abordagem estocástica e paralela permite escapar de ótimos locais, aumentando a probabilidade de encontrar soluções globalmente ótimas (Russell; Norvig, 2013).

Buscando aprimorar essa busca, surgem os Algoritmos Meméticos (MA), descritos por Moscato como uma abordagem que mimetiza a evolução cultural. Diferente dos AGs clássicos, os MAs combinam a busca global populacional com uma busca local heurística, onde cada indivíduo refina sua solução antes de interagir com os demais. Essa hibridização é especialmente eficaz em ambientes paralelos, onde a divisão de tarefas favorece a evolução de subpopulações.

Nesse contexto, os Algoritmos Genéticos Paralelos consolidaram-se como uma abordagem eficiente para acelerar o processo evolutivo dos AGs tradicionais, distribuindo a carga computacional entre múltiplos processadores. Uma estratégia amplamente utilizada é o modelo *coarse-grained*, também conhecido como modelo em ilhas, no qual a população é dividida em múltiplas subpopulações relativamente isoladas, que ocasionalmente trocam indivíduos. Essa estrutura favorece tanto a exploração diversificada do espaço de busca quanto ganhos substanciais de desempenho computacional (Cantú-Paz, 1998).

1.1 Justificativa

O avanço das arquiteturas paralelas tem possibilitado não apenas a redução do tempo de execução de algoritmos, mas também o desenvolvimento de estratégias evolutivas mais

eficazes para a obtenção de soluções de alta qualidade. No contexto do Problema de Roteamento de Veículos com Capacidade (CVRP), um problema clássico NP-difícil de grande relevância prática, métodos tradicionais baseados em Algoritmos Genéticos tendem a apresentar limitações relacionadas à convergência prematura e à exploração restrita do espaço de busca.

Nesse sentido, os Algoritmos Meméticos estruturados em modelo de ilhas destacam-se por permitir a evolução simultânea de múltiplas subpopulações, favorecendo a diversidade genética e o equilíbrio entre intensificação e diversificação da busca por meio da integração com operadores de busca local. Assim, o paralelismo adotado neste trabalho é explorado principalmente como uma estratégia algorítmica para melhorar a qualidade das soluções obtidas para o CVRP, enquanto a análise de métricas como *Speedup* e Eficiência é considerada de forma complementar, com o objetivo de caracterizar o comportamento computacional da abordagem proposta.

1.2 Objetivo geral

Investigar e implementar Algoritmos Meméticos Paralelos, estruturados em modelo de ilhas, aplicados ao Problema de Roteamento de Veículos com Capacidade (CVRP), com foco na obtenção de soluções de maior qualidade por meio do aumento da diversidade populacional e da incorporação de mecanismos de busca local, analisando de forma complementar o comportamento computacional da abordagem paralela.

1.3 Objetivo específico

- Estudar os fundamentos dos Algoritmos Meméticos e das estratégias de paralelismo em modelo de ilhas (*coarse-grained*), com ênfase nos mecanismos de diversificação e intensificação da busca.
- Implementar um Algoritmo Memético Sequencial para o CVRP, incorporando o operador de busca local 2-opt.
- Desenvolver uma versão paralela do algoritmo utilizando a biblioteca OpenMP, estruturada em modelo de ilhas com estratégias heterogêneas de evolução.
- Analisar o impacto de diferentes topologias de migração (Anel e Malha) na diversidade populacional, na convergência e na qualidade das soluções obtidas.
- Avaliar a qualidade das soluções por meio de métricas como *Gap* em relação a instâncias da

literatura, considerando adicionalmente métricas de desempenho paralelo, como *Speedup* e Eficiência, de forma complementar.

1.4 Estrutura do trabalho

O restante deste trabalho está organizado da seguinte forma:

- **Seção 2 – Fundamentação Teórica**: discute os conceitos essenciais necessários para o entendimento do trabalho, incluindo a definição formal do CVRP, os princípios de algoritmos genéticos, estratégias de paralelização com foco no modelo em ilhas e a utilização da biblioteca OpenMP para multiprocessamento.
- **Seção 3 – Trabalhos Relacionados**: apresenta uma revisão de pesquisas anteriores que abordam a aplicação de algoritmos genéticos, com destaque para abordagens que utilizaram modelos em ilhas ou técnicas similares. O capítulo serve como base comparativa para a proposta desenvolvida neste trabalho.
- **Seção 4 – Metodologia**: descreve de forma detalhada os métodos adotados para implementação do algoritmo genético sequencial e da versão paralela com modelo em ilhas. Inclui a modelagem do problema, estrutura dos indivíduos, operadores genéticos utilizados, critérios de parada, parametrização, ferramentas de desenvolvimento e ambiente de execução.
- **Seção 5 – Resultados**: apresenta os resultados obtidos com a aplicação da abordagem proposta. São analisados os ganhos de desempenho decorrentes da paralelização, a qualidade das soluções geradas e a eficiência do modelo em ilhas em comparação com o modelo sequencial, com base nos experimentos realizados.
- **Seção 6 – Conclusão**: sintetiza as principais contribuições do trabalho, discutindo os resultados alcançados, as limitações identificadas e as conclusões obtidas a partir dos experimentos.

2 FUNDAMENTAÇÃO

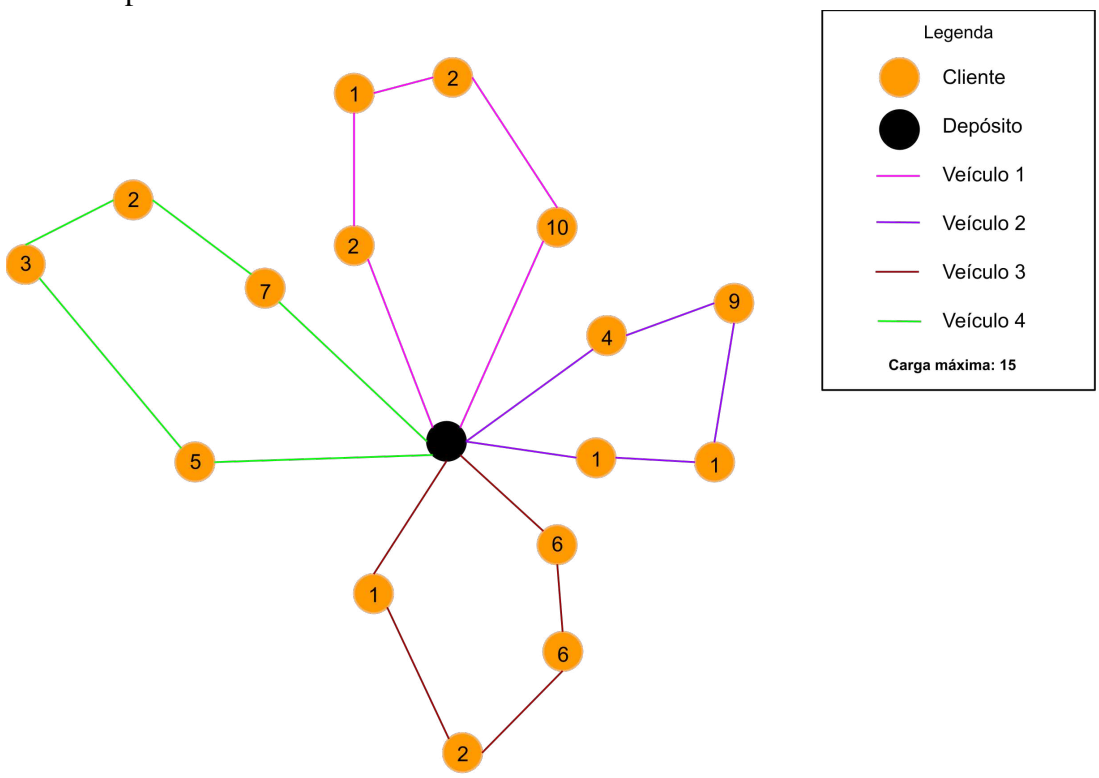
2.1 O Problema de Roteamento de Veículos com Capacidade (CVRP)

O *Vehicle Routing Problem* (VRP) teve sua origem formal em 1959, com o trabalho seminal de Dantzig e Ramser (1959), intitulado *The Truck Dispatching Problem*. Nele, os autores propuseram um modelo matemático para otimizar a distribuição de gasolina entre depósitos e postos de abastecimento, com o objetivo de minimizar a distância total percorrida pelos caminhões. Esse trabalho tornou-se a base conceitual para uma ampla gama de problemas logísticos de roteamento, dentre os quais o *Capacitated Vehicle Routing Problem* (CVRP) se destacou como uma das variações mais estudadas e aplicadas na prática (Dantzig; Ramser, 1959).

De acordo com Toth *et al.* (2014), o CVRP consiste na tarefa de transportar e distribuir mercadorias a partir de um depósito central (vértice 0) para um conjunto de clientes $N = \{1, 2, ..., n\}$, onde cada cliente $i \in N$ possui uma demanda $d_i > 0$. Considera-se uma frota homogênea de veículos $K = \{1, 2, ..., |K|\}$, todos localizados inicialmente no depósito. Cada veículo possui capacidade máxima $Q > 0$ e deve percorrer uma subrota $S \subseteq N$, atendendo às respectivas demandas sem ultrapassar sua capacidade, retornando ao depósito ao final do percurso. Cada par de localidades (i, j) é associado a um custo de viagem c_{ij}, tipicamente proporcional à distância entre os pontos.

O objetivo do CVRP é encontrar o conjunto de rotas de menor custo total, de forma que todos os clientes sejam atendidos exatamente uma vez, a capacidade de cada veículo não seja excedida e todas as rotas comecem e terminem no depósito, como demonstrado na Figura 1. Essa formulação adiciona restrições ao VRP tradicional, tornando o problema ainda mais complexo e desafiador do ponto de vista computacional (Toth *et al.*, 2014).

Figura 1 – Exemplo ilustrativo de uma instância do CVRP com 16 clientes e 4 veículos. As rotas coloridas representam o caminho percorrido por cada veículo, respeitando sua capacidade máxima.



Fonte: Elaboração do autor.

O CVRP possui inúmeras aplicações práticas, que vão além da simples entrega de mercadorias. Entre elas, destacam-se: coleta de resíduos sólidos, limpeza de ruas, roteirização de ônibus escolares, transporte sob demanda (como em aplicativos de mobilidade urbana), transporte de pessoas com deficiência, roteirização de vendedores e planejamento de rotas para equipes de manutenção (Toth; Vigo, 2002).

A dificuldade em obter soluções exatas para esses problemas reside no fato de que o CVRP é um problema **NP-difícil no sentido forte**, pois generaliza o clássico *Problema do Caixeiro Viajante* (TSP) — também pertencente à mesma classe de complexidade. Isso implica que não se conhece nenhum algoritmo de tempo polinomial capaz de resolver todos os casos do CVRP de forma ótima. Diante disso, o uso de *metaheurísticas* torna-se uma abordagem fundamental. Entre as mais utilizadas estão: Busca Tabu (*Tabu Search*), Têmpera Simulada (*Simulated Annealing*) e Algoritmos Genéticos, que visam encontrar soluções de alta qualidade que sejam computacionalmente factíveis (Toth; Vigo, 2002).

2.2 Algortimos Genéticos

Segundo Kumar *et al.* (2010), os Algoritmos Genéticos (AGs) consistem em um método de busca heurística adaptativa, inspirado nos princípios da genética de populações e nos mecanismos da seleção natural. Essa abordagem foi introduzida por John Holland no início da década de 1970, com o objetivo de resolver problemas complexos de otimização por meio de um processo evolutivo artificial.

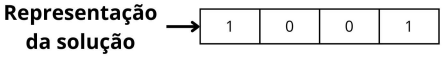
A lógica do algoritmo baseia-se na evolução de um conjunto de soluções candidatas, denominado população, em que cada solução é representada por um **cromossomo**, o qual corresponde à codificação completa de uma possível solução do problema. Cada cromossomo é formado por **genes**, que representam as menores unidades de informação, geralmente associadas às variáveis ou parâmetros do problema a ser otimizado. A cada geração, os indivíduos são avaliados com base em uma função de aptidão (*fitness*), que determina a qualidade de cada solução. Com base nesses valores, são selecionados probabilisticamente os cromossomos mais aptos para reprodução, resultando na geração de novos indivíduos por meio das operações de cruzamento (*crossover*) e mutação, que atuam diretamente sobre os genes. Esse processo visa garantir que as novas populações apresentem, em média, melhor desempenho do que as anteriores, promovendo uma evolução contínua das soluções ao longo do tempo (Kumar *et al.*, 2010). Nas subseções seguintes será discutido de forma mais detalhada cada etapa dos algoritmos genéticos.

2.2.1 Inicialização

A etapa de inicialização consiste na criação da população inicial do problema. Essa população é composta por um conjunto de indivíduos, cada um representando uma possível solução válida. Para cada indivíduo da população, é necessário definir uma representação adequada, que pode variar conforme o tipo de problema a ser resolvido. Como mostrado por Eiben e Smith (2003), existem algumas representações básicas utilizadas em Algoritmos Genéticos, cada uma mais apropriada para determinados tipos de problemas.

A primeira delas é a **representação binária**, na qual os cromossomos são formados por sequências de *bits* (0s e 1s). Essa representação é particularmente eficaz para problemas de decisão booleana, como o Problema da Mochila, onde o valor 1 pode indicar que um item será incluído na mochila, enquanto o valor 0 indica que ele será descartado (Eiben; Smith, 2003).

Figura 2 – Exemplo de representação binária aplicado ao problema da mochila.

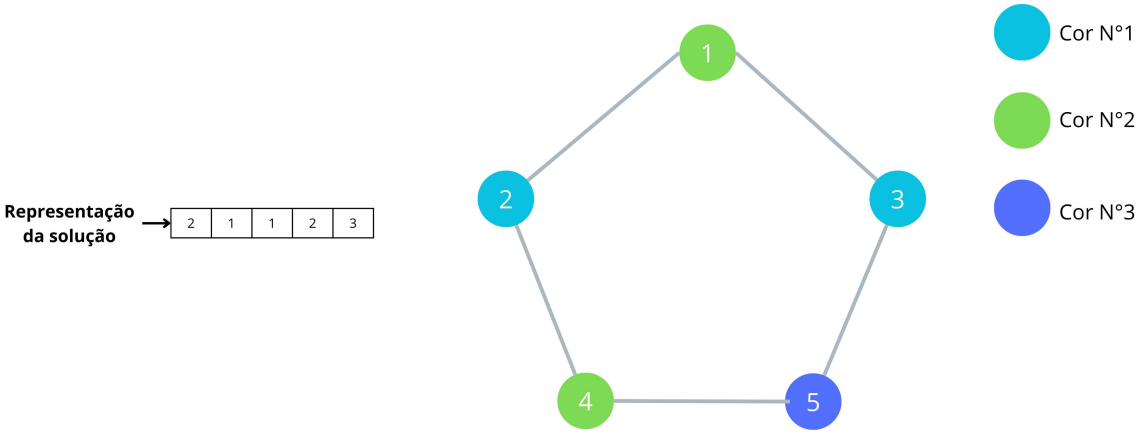


ITEM	PESO	IMPORTÂNCIA
ITEM 1	10	6
ITEM 2	20	1
ITEM 3	5	5
ITEM 4	10	2

Fonte: Adaptada de Eiben e Smith (2003).

Outra forma comum é a **representação inteira**, na qual os genes são valores inteiros. Esse tipo de codificação é útil para problemas onde as variáveis assumem valores discretos, como no problema de coloração de grafos. Nesse problema, dado um conjunto de vértices e suas conexões (arestas), o objetivo é atribuir uma das k cores a cada vértice, de modo que vértices adjacentes não compartilhem a mesma cor (Eiben; Smith, 2003).

Figura 3 – Exemplo de representação inteira aplicada à coloração de grafos.



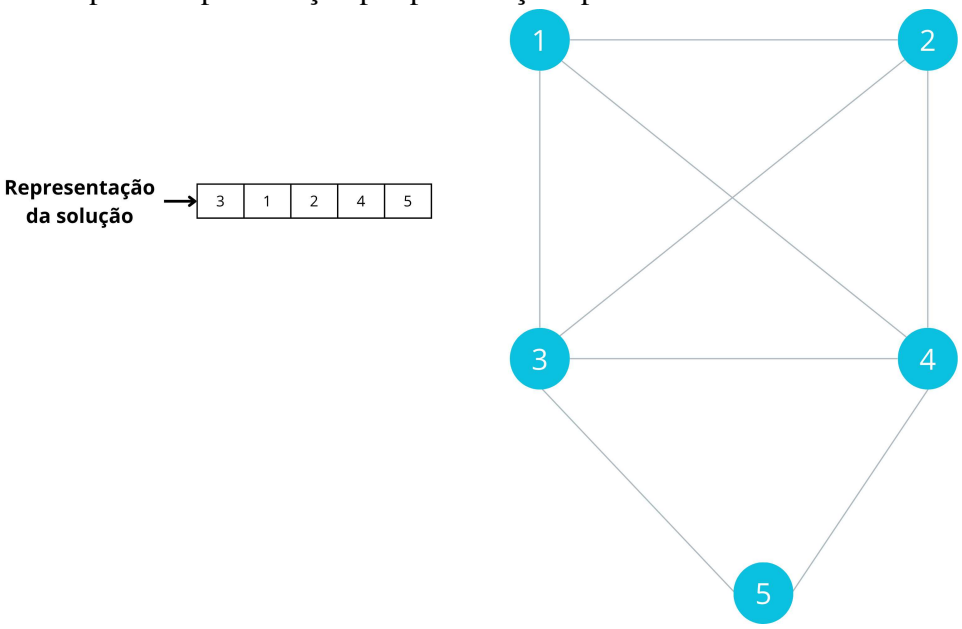
Fonte: Adaptada de Eiben e Smith (2003).

Há também a **representação por permutação**, especialmente adequada para problemas de ordenação e roteamento, como o Problema do Caixeiro Viajante (TSP). Nesse caso, o cromossomo representa uma permutação dos elementos, indicando a ordem em que as cidades devem ser visitadas. Essa representação exige operadores genéticos específicos, como o crossover por ordenação (*Order Crossover* – OX) e mutações baseadas em trocas, para garantir que as soluções permaneçam válidas (sem cidades repetidas ou ausentes) (Eiben; Smith, 2003).

A escolha da representação influencia diretamente o desempenho do algoritmo e define quais operadores genéticos podem ser aplicados com eficácia (Eiben; Smith, 2003).

Em geral, esses indivíduos são gerados de forma aleatória, promovendo uma diversi-

Figura 4 – Exemplo de representação por permutação aplicada ao TSP.



Fonte: Adaptada de Eiben e Smith (2003).

dade inicial ampla. No entanto, dependendo do problema, é possível utilizar abordagens mais dirigidas, baseadas em conhecimento prévio sobre o espaço de busca, com o objetivo de concentrar as soluções iniciais em regiões com maior probabilidade de conter soluções ótimas (Kumar *et al.*, 2010).

2.2.2 Seleção

A seleção é a etapa responsável por escolher os indivíduos que participarão do processo de reprodução. Nesse estágio, ainda não há criação de novos indivíduos; o foco está em preservar os mais promissores da população atual (Kumar *et al.*, 2010). Os métodos de seleção podem variar conforme a natureza do problema. A forma mais direta baseia-se na função de aptidão (*fitness*), que avalia a qualidade de cada indivíduo, permitindo que os mais aptos sejam escolhidos para a reprodução. Além disso, métodos estocásticos, como a seleção por roleta (*roulette wheel selection*) e a seleção por torneio (*tournament selection*), introduzem aleatoriedade ao processo, contribuindo para a manutenção da diversidade genética e evitando a convergência prematura para soluções subótimas (Shapiro, 2001)

Conforme discutido por Schmitt (2001), os principais métodos de seleção, e seu funcionamento, são os seguintes:

Seleção Proporcional à Aptidão:

Neste método, cada indivíduo tem uma chance de ser selecionado proporcional ao

seu valor de *fitness*. Suponha uma população $p = \{c_1, c_2, ..., c_n\}$, onde cada indivíduo $c_i \in p$ possui uma função de aptidão $f(c_i, p)$. A probabilidade de seleção de um indivíduo c^* é dada por: $P(c^*) = \frac{f(c^*,p)}{\sum_{i=1}^{n} f(c_i,p)}$. Caso todos os indivíduos tenham o mesmo valor de *fitness*, ou a soma total seja zero, aplica-se uma normalização na qual todos recebem *fitness* igual. Esse método apresenta limitações quando há pouca variabilidade entre os indivíduos ou quando um único indivíduo domina a população, levando à perda rápida de diversidade (Schmitt, 2001).

Seleção por Torneio:

Nesse método, dois ou mais indivíduos são escolhidos aleatoriamente da população para competir em um "torneio". A seleção do vencedor pode ser determinística (o mais apto vence sempre) ou probabilística. Em sua forma estocástica, é definida uma variável $G \in [0, 0.5)$, tal que:

- O melhor entre os indivíduos sorteados é selecionado com probabilidade $1 - G$;
- O pior é selecionado com probabilidade G.

Valores de G mais próximos de 0.5 aumentam a chance de indivíduos menos aptos serem escolhidos, promovendo diversidade, enquanto valores próximos de 0 reduzem essa chance (Schmitt, 2001).

Seleção por *Ranking*:

Diferentemente da seleção proporcional, a seleção por *ranking* ordena os indivíduos com base em sua classificação (*ranking*) de *fitness*, e não em seus valores absolutos. Assim, a probabilidade de seleção depende da posição do indivíduo na ordenação, o que evita problemas com valores extremos de *fitness* e assegura uma pressão seletiva mais controlada (Schmitt, 2001).

2.2.3 Reprodução

A reprodução é essencial para a formação de novas gerações. Durante essa etapa, dois indivíduos da população atual são selecionados para gerar um novo descendente, por meio do cruzamento de suas informações genéticas. Esse processo é repetido até que a nova população atinja o tamanho previamente definido. O objetivo é transmitir e combinar características vantajosas dos pais, promovendo a evolução da população e, consequentemente, a melhoria da qualidade das soluções ao longo das gerações (Kumar *et al.*, 2010).

Existem diferentes formas de se realizar essa combinação de indivíduos dentre elas temos as seguintes:

2.2.3.1 Crossover Parcialmente Mapeado (Partially Mapped Crossover ou PMX):

A operação inicia-se com a seleção de dois pontos de corte aleatórios no cromossomo. O segmento compreendido entre esses pontos é copiado diretamente do indivíduo 1 para a prole, preservando as posições originais dos genes. Em seguida, os genes que ainda não estão presentes na prole são inseridos com base no indivíduo 2, por meio de um processo de mapeamento que evita duplicações. Para cada gene ausente a ser transferido, verifica-se a posição ocupada, no indivíduo 1, pelo gene que corresponde à mesma posição no indivíduo 2. O gene ausente é então alocado na prole na posição associada a esse gene do indivíduo 1. Caso essa posição já esteja ocupada, o procedimento é repetido de forma iterativa, seguindo o mapeamento entre os genes dos dois indivíduos, até que uma posição livre seja encontrada. Esse processo garante que cada gene apareça exatamente uma vez na prole, preservando a consistência da permutação (Eiben; Smith, 2003).

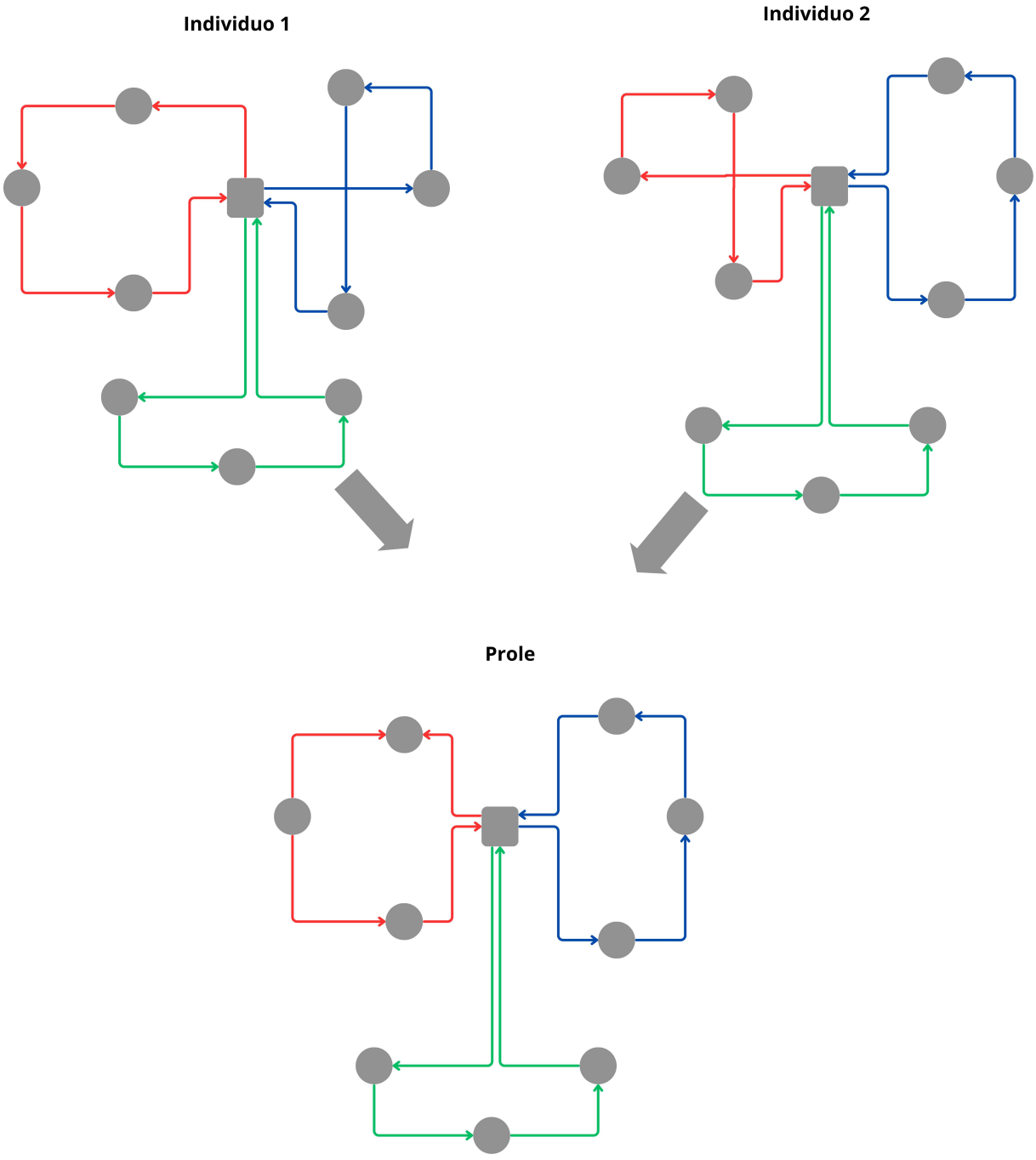
Figura 5 – Exemplo do funcionamento PMX.

Fonte: Adaptada de Eiben e Smith (2003).

2.2.3.2 Crossover Baseado em Rotas (Route-Based Crossover ou RBX:)

Esse operador cria uma nova prole por meio da substituição de rotas. Inicialmente, considerando o indivíduo 1, uma de suas rotas é selecionada e, em seguida, essa rota é substituída por uma rota proveniente do indivíduo 2, dando origem a uma nova solução descendente. Contudo, devido à natureza desse operador, é possível que ocorram clientes duplicados ou rotas que ultrapassem a capacidade máxima permitida para cada veículo. Dessa forma, torna-se necessário que essas soluções sejam corrigidas antes de serem incorporadas à geração seguinte (Potvin; Bengio, 1996).

Figura 6 – Exemplo do funcionamento RBX.



Fonte: Adaptado de Potvin e Bengio (1996).

2.2.4 Mutação

A mutação é uma operação fundamental nos algoritmos genéticos, pois tem como objetivo preservar a diversidade genética ao longo das gerações. Essa operação consiste em reali-

zar pequenas alterações aleatórias nos genes de um indivíduo, de modo a evitar a convergência prematura para ótimos locais e possibilitar a exploração de novas regiões do espaço de busca. Em geral, a mutação é aplicada com baixa probabilidade, atuando como um mecanismo de perturbação que rompe padrões homogêneos na população e amplia a capacidade exploratória do algoritmo. O Quadro 1 apresenta os principais operadores de mutação utilizados neste trabalho, bem como suas características gerais, conforme discutido por Mathew (2003).

Quadro 1 – Alguns tipos de operadores de mutação

Mutação	**Descrição**
Swap Mutation	Seleciona aleatoriamente dois genes no cromossomo e troca seus valores. Exemplo: trocar os genes na posição 2 e 5.
Insert Mutation	Dois genes são selecionados, e o segundo é movido para a posição imediatamente após o primeiro, reorganizando os demais para abrir espaço. Exemplo: mover o gene na posição 5 para após o gene na posição 2.
Scramble Mutation	Uma subsequência do cromossomo (ou o cromossomo inteiro) tem suas posições embaralhadas aleatoriamente. Exemplo: embaralhar os genes da posição 2 a 5.
Bit Flip Mutation	Para codificações binárias, cada gene tem uma probabilidade p_m de ser invertido (0 para 1 ou 1 para 0). A quantidade de genes alterados segue uma distribuição binomial com média $L \cdot p_m$, onde L é o comprimento do cromossomo.

Fonte: Adaptada de Eiben e Smith (2003)

2.2.5 Critérios de Parada

Segundo Shapiro (2001), os algoritmos genéticos seguem seu ciclo evolutivo iterativamente até que um critério de parada seja satisfeito — como, por exemplo, o alcance de um número máximo de gerações ou a descoberta de uma solução com qualidade considerada aceitável.

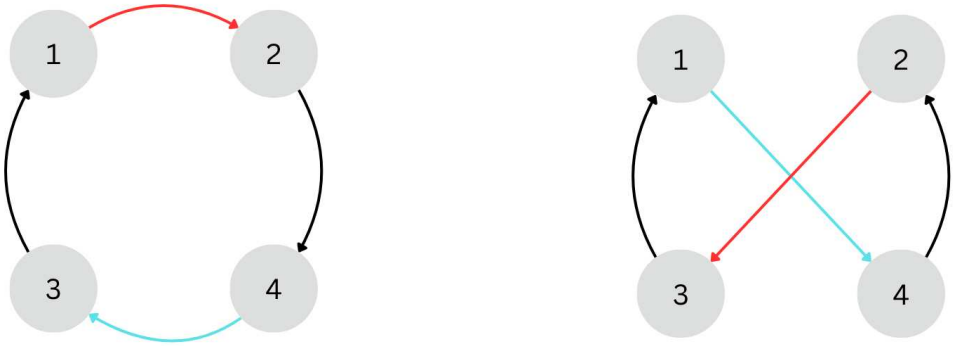
2.3 Busca Local e o Operador 2-Opt

A busca local é uma estratégia amplamente utilizada em problemas de otimização combinatória, como os VRSPs (Vehicle Routing and Scheduling Problems). A ideia central da busca local é partir de uma solução inicial e, em seguida, explorar soluções vizinhas por meio da operação simples de modificar parcialmente essa solução inicial, obtendo assim novas soluções vizinhas. Em cada iteração da busca local, algumas ou todas as soluções vizinhas são avaliadas;

caso exista uma solução com custo menor que o da solução atual, o algoritmo se move para essa nova solução. Esse procedimento se repete até que não seja possível encontrar nenhuma solução vizinha que melhore o custo. Quando isso ocorre, dizemos que a solução atual é um ótimo local em relação à vizinhança considerada, e o processo de busca local é encerrado (Funke *et al.*, 2005).

Entre os algoritmos de busca local, um dos mais simples é o 2-opt, proposto inicialmente por Croes (1958). Contudo, o movimento básico já havia sido descrito e sugerido anteriormente por Flood (1956). A operação consiste em eliminar duas arestas de um percurso, dividindo-o em duas partes, e então reconectar esses caminhos de uma forma diferente da original, buscando assim uma solução de menor custo (Johnson; McGeoch, 1997).

Figura 7 – Exemplo do funcionamento 2-Opt.



Fonte: Adaptado de Johnson e McGeoch (1997).

2.4 Algoritmos Meméticos

Os algoritmos meméticos são híbridos que combinam a busca global populacional, característica dos algoritmos genéticos, com técnicas de busca local, como *hill climbing* e 2-opt. Diferentemente do AG puro, que depende exclusivamente de operadores genéticos — seleção, cruzamento e mutação — o algoritmo memético incorpora um mecanismo de aprendizado individual ao longo do processo evolutivo. Dessa forma, cada indivíduo pode se autoaprimorar durante sua evolução (Moscato, 1989).

A base do conceito "memético" está no termo "meme", introduzido por Dawkins (1976), que define meme como uma unidade de transmissão cultural, análoga ao gene na biologia. No contexto de algoritmos genéticos, isso significa que o indivíduo não apenas herda informação por meio dos genes, mas também refina seu próprio comportamento através da busca local, representando um processo de aprendizado cultural (Moscato, 1989).

O ponto central da aplicação da busca local é equilibrar exploração e intensificação, algo que o algoritmo genético puro apresenta maior dificuldade em alcançar de forma eficiente. A ideia é que os indivíduos não dependam exclusivamente dos genes herdados; assim, eles são imediatamente melhorados para participar das próximas gerações. Como consequência, o cruzamento passa a ser realizado entre soluções já otimizadas, aumentando a eficiência do processo evolutivo (Moscato, 1989).

2.5 Paralelismo

Nesta seção são apresentados os principais conceitos relacionados ao paralelismo computacional, abordando inicialmente as diferentes arquiteturas paralelas, com ênfase na organização da memória e na forma de comunicação entre os processadores. Em seguida, são discutidas as principais métricas de desempenho utilizadas na avaliação de algoritmos paralelos, incluindo speedup, eficiência, a Lei de Amdahl e escalabilidade, as quais permitem analisar o impacto do paralelismo no tempo de execução e no aproveitamento dos recursos computacionais.

2.5.1 Arquitetura

Segundo Rauber e Rünger (2010), as arquiteturas paralelas podem ser distinguidas de acordo com a organização física da memória. Elas podem ser fisicamente compartilhadas, também chamadas de *multiprocessors*, constituídas por vários processadores ou núcleos que compartilham uma única memória, denominada memória global. Essa organização permite que os processadores troquem informações por meio de variáveis compartilhadas presentes nessa memória comum. Um exemplo típico desse modelo são os núcleos de um processador multicore.

Outra forma de organização é a memória fisicamente distribuída, também conhecida como arquitetura *multicomputer*, composta por diversos elementos, denominados nós, interconectados de modo a possibilitar a comunicação e a troca de informações entre eles. Cada nó é independente dos demais, possuindo memória própria, processador próprio e, em alguns casos, seus próprios periféricos (Rauber; Rünger, 2010).

2.5.2 Métricas de desempenho

Para que a avaliação do desempenho de um software paralelo em comparação com sua versão sequencial seja feita de forma coerente e válida, é necessário adotar métodos

padronizados de medição. Nesse contexto, surgem métricas específicas voltadas ao cálculo de desempenho de algoritmos paralelos, as quais permitem analisar de maneira geral o quão eficiente foi a execução paralela, considerando fatores como ganho de tempo, escalabilidade e aproveitamento dos recursos computacionais (Rauber; Rünger, 2010).

2.5.2.1 Speedup

Para a validação de um algoritmo paralelo em comparação com sua implementação sequencial, é de extrema importância analisar o tempo de execução de ambas as abordagens, a fim de identificar os benefícios proporcionados pelo paralelismo. De modo geral, essa comparação é realizada por meio da economia relativa no tempo de execução, expressa pelo conceito de speedup (Rauber; Rünger, 2010). Esse conceito é representado pela seguinte fórmula:

$$S_p(n) = \frac{T^*(n)}{T_p(n)}$$

Onde n representa o **tamanho do problema** (por exemplo, a quantidade de dados de entrada), $T^*(n)$ é o **tempo de execução do algoritmo sequencial**, e $T_p(n)$ é o **tempo de execução do algoritmo paralelo** utilizando p processadores para resolver o mesmo problema. Ao final, podemos observar o **ganho de desempenho**, ou seja, a economia relativa de tempo obtida ao comparar a execução do algoritmo sequencial com a execução paralela utilizando p processadores.

2.5.2.2 Eficiência

Outra medida alternativa para o cálculo de desempenho de um programa paralelo é a eificiência. Ela consiste basicamente na fração de tempo em que o processador passou realmente trabalhando. Assim como o speedup, tambem é necessário a execução do algoritmo sequencial (Rauber; Rünger, 2010). Podemos definir como:

$$E_p(n) = \frac{S_p(n)}{p}$$

Sendo $S_p(n)$ o speedup com p processadores, e p o número de processadores usados para resolver a entrada n.

2.5.2.3 *Lei de Amdahl*

Entre os ganhos possíveis, existem limitadores que podem restringir esses benefícios, como, por exemplo, o número de processadores. No entanto, há também uma outra restrição importante, que surge com base nas partes de um programa que podem ser paralelizadas e nas partes que precisam ser executadas de forma sequencial. Essa influência pode ser facilmente representada pela Lei de Amdahl (Rauber; Rünger, 2010). A estimativa é dada pela função:

$$S_p(n) = \frac{T^*(n)}{(1-f)\,T^*(n) + \frac{f\,T^*(n)}{p}} = \frac{1}{(1-f) + \frac{f}{p}} \leq \frac{1}{1-f}, \tag{2.1}$$

Essa estimativa pressupõe que o melhor algoritmo sequencial é utilizado e que a parte paralela do programa é perfeitamente paralelizada. O efeito da fração sequencial limita o speedup máximo que pode ser alcançado: se, por exemplo, 20% do programa precisa ser executado sequencialmente ($f = 0.2$), então o speedup máximo teórico é

$$S_{\text{máx}} = \frac{1}{f} = \frac{1}{0.2} = 5,$$

independentemente do número de processadores utilizados. Portanto, partes do programa que devem ser executadas sequencialmente devem ser levadas em consideração, especialmente quando um grande número de processadores é empregado (Rauber; Rünger, 2010).

2.5.2.4 *Escalabilidade*

A escalabilidade consiste em avaliar se é possível alcançar melhorias de desempenho proporcionais ao aumento do número de processadores utilizados, sendo esta métrica dependente de diversas propriedades do algoritmo e de sua execução paralela; a ideia é verificar, para um problema de tamanho fixo n, se o ganho de desempenho acompanha o aumento do número p de processadores, demonstrando que, com um número suficientemente grande de processadores, é possível resolver problemas grandes em um tempo comparável ao de problemas pequenos (Rauber; Rünger, 2010).

2.6 OpenMP

O OpenMP é um padrão portátil para programação paralela em sistemas de memória compartilhada. Ele fornece um conjunto de diretivas de compilador, rotinas de biblioteca e variáveis de ambiente que permitem estender linguagens originalmente sequenciais, como Fortran, C e C++, para a execução paralela (Rauber; Rünger, 2010).

O modelo de programação do OpenMP baseia-se na execução cooperativa de *threads* que operam simultaneamente em múltiplos processadores ou núcleos. A criação e a finalização dessas *threads* seguem o paradigma de bifurcação e junção (*fork-join*), no qual um programa OpenMP inicia sua execução com uma única *thread*, responsável por executar o código de forma sequencial. Ao alcançar uma região paralela, essa *thread* inicial cria um conjunto adicional de *threads*, formando uma equipe que passa a executar o código paralelamente. Ao final da região paralela, as *threads* são sincronizadas e encerradas, retornando a execução para a *thread* principal (Rauber; Rünger, 2010).

O trecho de código delimitado por uma construção paralela é denominado região paralela e é executado simultaneamente por todas as *threads* da equipe. O modo de execução pode seguir o paradigma SPMD (*Single Program, Multiple Data* — Programa Único, Múltiplos Dados), no qual múltiplas instâncias do mesmo código operam sobre diferentes partes dos dados. Alternativamente, o OpenMP também permite a atribuição de tarefas distintas a *threads* diferentes, proporcionando maior flexibilidade no processo de paralelização (Rauber; Rünger, 2010).

No que diz respeito ao modelo de memória, o OpenMP distingue entre memória compartilhada e memória privada. Todas as *threads* possuem acesso ao espaço de memória compartilhada, enquanto variáveis privadas são exclusivas de cada *thread*. Dessa forma, torna-se necessário o uso de mecanismos de sincronização para evitar conflitos de acesso, como condições de corrida, garantindo a corretude e a consistência da execução paralela (Rauber; Rünger, 2010).

2.7 Algoritmos Genéticos Paralelos

Embora os Algoritmos Genéticos (AGs) produzam resultados positivos em várias categorias de problemas, sua performance pode ser significativamente afetada quando utilizados em problemas de grande escala e elevada complexidade computacional. Conforme o espaço de pesquisa e a quantidade de variáveis aumentam, o tempo necessário para encontrar soluções

viáveis também se amplia, tornando-se um obstáculo em aplicações práticas (Cantú-Paz, 1998).

Diante desse cenário, surgem os Algoritmos Genéticos Paralelos (PGAs, do inglês *Parallel Genetic Algorithms*) como uma abordagem promissora para melhorar tanto a eficiência quanto a eficácia da busca evolutiva. Diferentemente de uma simples versão paralela dos AGs sequenciais, os AGPs formam uma classe de metaheurísticas que, ao incorporar execução paralela e estruturas populacionais organizadas, podem explorar melhor o espaço de busca e preservar a diversidade genética ao longo das gerações (Cantú-Paz, 1998; Alba; Troya, 1999).

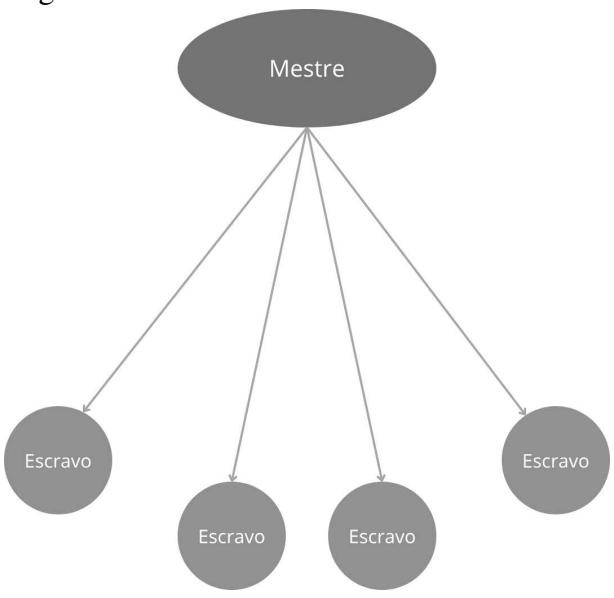
Além de possibilitar ganhos expressivos de desempenho computacional, os AGPs oferecem características desejáveis como afirmado por Alba e Troya (1999):

- Intensidade e adaptabilidade na representação;
- Procura simultânea de vários pontos no espaço de soluções;
- Aprimoramento na exploração e preservação da diversidade;
- Habilidade para produzir diversas soluções alternativas.

Os AGPs são tradicionalmente classificados em três principais modelos, como demonstrado e explicado por Cantú-Paz (1998):

- **Modelo *Master-Slave***: São semelhantes aos AGs sequenciais, porém a etapa de avaliação da aptidão é distribuída entre diversos processadores (escravos), enquanto outro processador executa as operações do AG e realiza a distribuição dos indivíduos;

Figura 8 – Modelo *master-slave*.

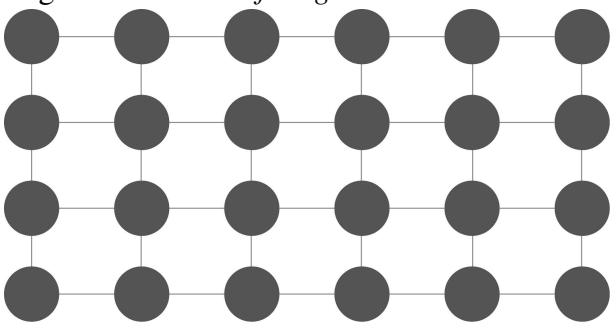


Fonte: Adptada de Cantú-Paz (1998).

- **Modelo *Fine-Grained***: São adequadas para computadores massivamente paralelos, nos

quais existe apenas uma única população, e as operações de crossover e seleção ocorrem somente entre vizinhos próximos.;

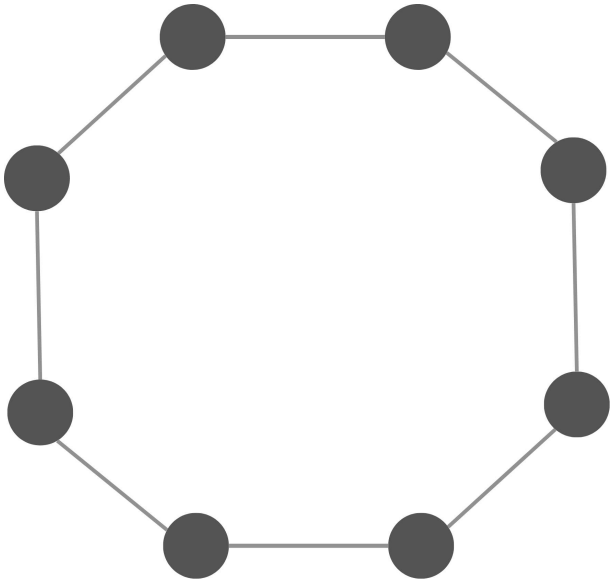
Figura 9 – Modelo *fine-grained*.



Fonte: Adptada de Cantú-Paz (1998).

- **Modelo *Coarse-Grained* (ou Modelo em Ilhas)**: Um dos modelos mais populares, no qual múltiplas subpopulações (ilhas) evoluem de forma relativamente independente, realizando trocas ocasionais de indivíduos por meio de mecanismos de migração.

Figura 10 – Modelo multiplas populações.



Fonte: Adptada de Cantú-Paz (1998).

Neste trabalho, será dada ênfase ao Modelo em Ilhas, dada sua aplicabilidade prática, flexibilidade e bons resultados em problemas combinatórios. Uma descrição mais aprofundada desse modelo será apresentada na próxima seção.

2.8 Modelo em Ilhas

Os AGPs no modelo *Coarse-Grained* (ou Modelo em Ilhas) consistem em múltiplas populações divididas em ilhas (ou *demes*), que trocam indivíduos periodicamente por meio de um processo denominado migração. Cada ilha executa sua própria AG sequencial com uma população de tamanho reduzido, sendo comum que, em determinados momentos, selecione e envie seu melhor indivíduo para migração, por exemplo (Cantú-Paz, 1998).

Existem dois tipos comuns de migração como demostra Cantú-Paz (1998):

- **Migração síncrona**, que ocorre em intervalos de tempo definidos (sendo a mais comum entre os AGPs);
- **Migração assíncrona**, na qual a comunicação entre as ilhas acontece apenas após um evento específico — como o início da convergência da população ou após um número determinado de gerações.

Já a seleção dos indivíduos para migração pode ocorrer de diversas formas: pode-se optar por enviar o melhor indivíduo da população (como já mencionado) ou selecionar um indivíduo aleatório. Esta última abordagem é recomendada quando não se possui conhecimento prévio suficiente sobre o problema, pois reduz o risco de convergência prematura entre as ilhas. Além disso, tem-se também o conceito de **taxa de migração**, que consiste no número de indivíduos que serão transferidos entre as ilhas durante o processo de migração. Esse valor pode ser definido em termos percentuais (proporção da população total) ou como um número absoluto de indivíduos (Alba; Troya, 1999).

Embora este trabalho adote uma estratégia de migração periódica tradicional, vale mencionar abordagens mais recentes que exploram critérios adaptativos para o controle da migração. Um exemplo é o trabalho proposto por Li e Huang (2012), que, em vez de trabalhar com intervalos fixos de migração, apresenta a opção de migrações adaptativas: o AMDPGA (*Adaptive Migration Strategy of Distributed Parallel Genetic Algorithms*), que se utiliza do Índice de Similaridade de Indivíduos (ISIM) para decidir dinamicamente quando e se a migração deve ocorrer. A ideia é que, por meio dessa estratégia, seja possível obter um ganho na eficiência e na taxa de convergência dos algoritmos individuais durante o processo de migração, além de reduzir os gastos com comunicação e sincronização provocados por migrações desnecessárias, garantindo que indivíduos de alta qualidade possam se espalhar com eficiência pelo sistema.

Outro fator importante a ser considerado é a **topologia das ilhas**, ou seja, o padrão de comunicação entre elas. Caso exista uma conectividade muito densa entre as ilhas, soluções

ótimas — especialmente quando se utiliza o envio dos melhores indivíduos — podem-se espalhar rapidamente, promovendo uma convergência precoce. Em contrapartida, topologias com baixa conectividade tornam as ilhas mais isoladas, o que favorece a diversidade e o surgimento de novas soluções. Além disso, ao definir a topologia, deve-se considerar o custo de comunicação entre as ilhas, que pode impactar diretamente o desempenho do sistema (Cantú-Paz, 1998).

A implementação de Algoritmos Genéticos Paralelos no modelo em ilhas envolve inúmeros desafios técnicos e estratégicos. Como demonstrado por Whitley *et al.* (1999), a diversidade genética de cada ilha pode depender significativamente das condições iniciais de sua população, influenciando diretamente as trajetórias evolutivas de cada subpopulação. Além disso, a frequência de migração afeta o equilíbrio entre diversidade e convergência: migrações muito frequentes podem levar as ilhas a convergir rapidamente para soluções subótimas compartilhadas, enquanto uma frequência bem ajustada favorece a exploração mais ampla do espaço de busca e a convergência para soluções de maior qualidade. Outro ponto relevante é o chamado efeito de *hitch-hiking*, que ocorre quando genes que compõem boas soluções parciais carregam, junto a si, partes do cromossomo com desempenho inferior, dificultando a recombinação eficaz em problemas não separáveis. Por fim, os autores também alertam que o aumento do número de subpopulações nem sempre resulta em ganhos evolutivos: a partir de certo ponto, os custos de comunicação, sincronização e montagem das soluções podem superar os benefícios da paralelização, impondo limites práticos à escalabilidade do modelo (Whitley *et al.*, 1999).

Embora a aplicação clássica do Modelo em Ilhas vise frequentemente a redução do tempo computacional via distribuição de carga, estudos recentes reforçam seu papel determinante na qualidade da solução final. Bevilaqua *et al.* (2019), ao aplicarem um Algoritmo Memético Baseado em Ilhas (IBMA) para problemas de roteamento complexos (2E-HVRP), demonstram que a arquitetura paralela atua como um mecanismo robusto de preservação da diversidade genética.

Segundo os autores, a estratégia de ilhas com migração de elite mostrou-se "sempre eficaz na melhoria da aptidão (fitness)" quando comparada a populações únicas, sendo crucial para minimizar a distância total das rotas e evitar a convergência prematura. Nessa perspectiva, o isolamento temporário das subpopulações permite que o algoritmo explore diferentes regiões do espaço de busca simultaneamente, enquanto a migração controlada injeta material genético promissor em ilhas estagnadas.

O presente trabalho adota uma abordagem análoga à de Bevilaqua *et al.* (2019),

utilizando o modelo em ilhas não apenas como um acelerador de hardware, mas como uma estratégia algorítmica fundamental para superar ótimos locais e alcançar soluções de qualidade superior às obtidas pelo modelo sequencial tradicional.

E embora o modelo de ilhas seja o mais conhecido e tradicional entre os algoritmos genéticos paralelos de múltiplas populações, há diversas formas de aplicar essa estratégia. Por exemplo, a pesquisa de Manyiel *et al.* (2021) utiliza uma arquitetura fundamentada no modelo mestre-escravo para a realização de múltiplas populações, onde cada subpopulação se concentra na otimização de um único propósito. Esta estratégia evidencia que, apesar do predomínio do modelo em ilhas, outras arquiteturas podem ser eficazes na promoção da diversidade e na melhoria da qualidade das soluções para problemas de roteamento.

3 TRABALHOS RELACIONADOS

Nesta seção iremos dissertar sobre alguns trabalhos com linhas de pesquisa simlares ao trabalho atual.

3.1 *Speedup vs. Quality: Asynchronous and Cluster-based Distributed Adaptive Genetic Algorithms for Ordered Problems*

O trabalho proposto por Ohira e Islam (2020) tem como objetivo analisar diferentes estratégias de paralelismo baseado no modelo em ilhas (*Island Model Genetic Algorithm* — IMGA), com foco em algoritmos adaptativos aplicados a problemas baseados em ordenação. O estudo avalia os impactos de diferentes tipos de migração — síncrona adaptativa, assíncrona adaptativa e uma abordagem baseada em *clusters* — tanto na qualidade das soluções quanto no ganho de desempenho computacional. Especificamente, os autores propõem múltiplas estratégias adaptativas: uma delas seleciona os melhores indivíduos para migrar com base em sua contribuição para a aptidão e diversidade (medida pela métrica de *Longest Common Subsequence* — LCS) , enquanto outra abordagem, mais distinta, utiliza o algoritmo *K-Means* para agrupar dinamicamente a população global em novas ilhas, também usando a distância LCS.

A implementação adota uma arquitetura de paralelismo distribuído baseada em *clusters*, na qual cada nó executa uma ilha de forma independente. Como um dos principais objetivos do trabalho era analisar o comportamento dessas estratégias, cada uma foi testada de forma isolada, permitindo uma análise detalhada de seus impactos.

Os resultados apresentados no trabalho analisado evidenciam que a utilização de algoritmos genéticos distribuídos, assíncronos e organizados em clusters permite alcançar ganhos significativos de desempenho computacional sem comprometer de forma substancial a qualidade das soluções. Os experimentos realizados com problemas ordenados demonstraram que as abordagens assíncronas reduzem o tempo total de execução e melhoram a escalabilidade do algoritmo quando comparadas a versões síncronas tradicionais. Além disso, a adaptação dinâmica dos parâmetros evolutivos mostrou-se fundamental para equilibrar o compromisso entre *speedup* e qualidade da solução, possibilitando diferentes configurações de acordo com as restrições de tempo e precisão exigidas. Os resultados indicam que a paralelização baseada em clusters, aliada à execução assíncrona, é particularmente eficaz em ambientes distribuídos, proporcionando uma utilização mais eficiente dos recursos computacionais e mantendo soluções competitivas em

relação às melhores abordagens centralizadas.

Diante disso, este trabalho diferencia-se ao aplicar o modelo em ilhas especificamente ao CVRP, investigando o impacto de diferentes topologias de migração fixas (anel e malha) — um aspecto não explorado experimentalmente no trabalho de Ohira e Islam (2020). Adicionalmente, utiliza-se uma arquitetura baseada em paralelismo por *threads* em memória compartilhada, que oferece uma solução mais acessível para ambientes *multicore*, contrastando com a abordagem distribuída em *clusters* adotada por eles, para mais detalhes consultar o Quadro 2.

Quadro 2 – Comparação entre o trabalho de Ohira e Islam (2020) e este trabalho

Critério	**Ohira & Islam (2020)**	**Este Trabalho**
Modelo de Paralelismo	Paralelismo distribuído (Modelo em Ilhas — *Cluster*)	Paralelismo com *threads* (memória compartilhada)
Topologia das Ilhas	Migração global com seleção adaptativa via diversidade LCS; no modelo com *clustering*, as ilhas são reagrupadas dinamicamente via *K-Means*.	Fixas: Anel e Malha
Critério de Migração	Baseado na similaridade dos indivíduos (distância LCS) e na aptidão	Seleção por Torneio
Tipo de Migração	Testadas separadamente: adaptativa síncrona, adaptativa assíncrona e adaptativa por *cluster*	Assíncrona com 15% de chance de ocorrer por geração
Taxa de Migração	Adaptativa, calculada dinamicamente com base na diversidade da população	Fixa em 5%
Critério de Parada	Número de gerações proporcional ao tamanho do problema ($n \times 100$)	Número fixo de gerações
Problema Abordado	Problemas de ordenação: TSP, CVRP e JSP	Problema de Roteamento de Veículos Capacitado (CVRP)
Foco Principal	Analisar o *trade-off* entre *speedup* e qualidade em diferentes tipos de migração adaptativa	Testar a viabilidade de encontrar boas soluções em tempo hábil via paralelismo multicore e busca local
Plataforma Computacional	*Cluster* (máquinas distribuídas)	CPU com paralelismo por *threads*

Fonte: Elaborada pelo autor

3.2 *Exploring dynamic population Island genetic algorithm for solving the capacitated vehicle routing problem*

A ideia proposta por Rezaei *et al.* (2024) consiste em uma abordagem híbrida baseada em Algoritmos Genéticos no modelo em ilhas com população dinâmica, aplicada diretamente ao problema do CVRP. Nessa proposta, as ilhas podem perder toda a sua população e, consequentemente, serem removidas do sistema durante o processo evolutivo, caracterizando um ambiente dinâmico de subpopulações. Além disso, cada ilha executa um algoritmo genético híbrido refinado (HGS-CVRP), que incorpora mecanismos de busca local, reinicialização e controle de diversidade, com o objetivo de melhorar a qualidade das soluções, alcançar um equilíbrio eficaz entre intensificação e diversificação, e proporcionar ganhos de desempenho computacional.

A metodologia adotada emprega uma estratégia de migração em que um indivíduo aleatório da ilha com melhor fitness médio (*IslandSolution*) é enviado para uma ilha escolhida aleatoriamente. Esse mecanismo de migração é **não conservativo**, pois o indivíduo migrante é removido da população da ilha de origem ao mesmo tempo em que é inserido na ilha de destino, diferentemente de modelos clássicos em que a migração ocorre por cópia. Dessa forma, ilhas com melhor desempenho passam a perder indivíduos progressivamente a cada evento de migração, reduzindo seu tamanho populacional ao longo da execução. A topologia definida no estudo é totalmente conectada, o que significa que qualquer ilha pode se comunicar diretamente com qualquer outra. Caso uma ilha não consiga repor seus indivíduos por meio dos operadores evolutivos e do mecanismo de reinicialização, sua população pode ser completamente esvaziada, levando à sua remoção automática do sistema e reforçando a natureza dinâmica da abordagem. Os operadores genéticos utilizados são o crossover do tipo OX (*Order Crossover*) e a utilização de operadores de busca local como *Swap*, *Swap** e *2-opt*, que contribuem para intensificar a exploração de soluções adequadas. O experimento foi conduzido com quatro ilhas, cada uma iniciando com 100 indivíduos.

Os resultados obtidos demonstram que o algoritmo proposto DPIGA-HGS, que combina um Algoritmo Genético em Ilhas com população dinâmica e uma busca genética híbrida como mecanismo de refinamento local, apresentou desempenho superior em relação aos métodos estado da arte na resolução do Problema de Roteamento de Veículos Capacitado (CVRP). Os experimentos realizados em benchmarks clássicos (Uchoa, CMT e Golden) e em instâncias reais (LoggiBUD) evidenciaram uma redução significativa dos *gaps* médio e máximo

em relação às melhores soluções conhecidas, bem como um aumento expressivo no número de *Best-Known Solutions* (BKS) encontradas. Em particular, o DPIGA-HGS superou algoritmos consolidados como HGS-CVRP, SISR, ICAHGS e OR-Tools, demonstrando maior eficiência tanto na qualidade das soluções quanto na exploração do espaço de busca. Esses resultados confirmam que a estratégia de população dinâmica, aliada a mecanismos eficazes de intensificação e reinicialização, contribui de forma relevante para o aprimoramento do desempenho na resolução do CVRP.

Em relação à presente pesquisa, destacam-se diferenças metodológicas importantes. Este trabalho adota um modelo em ilhas com populações fixas, sem remoção dinâmica, focando na análise do impacto das topologias de migração (anel, malha e aleatória) no desempenho e na qualidade das soluções, além de realizar uma comparação direta com um Algoritmo Genético sequencial, aspecto não explorado no trabalho de Rezaei *et al.* (2024). Outro diferencial é o uso de paralelismo por *threads*, uma solução mais acessível e eficiente em ambientes *multicore*. Dessa forma, este estudo busca compreender não apenas os ganhos de desempenho, mas também como diferentes estruturas de comunicação entre ilhas influenciam a diversidade e a qualidade das soluções no contexto do CVRP, para mais detalhes comparativos consultar o Quadro 3.

Quadro 3 – Comparação entre o trabalho de Rezaei *et al.* (2024) e este trabalho

Critério	**Rezaei *et al.* (2024)**	**Este Trabalho**
Modelo de Paralelismo	*Island Model* com população dinâmica	*Island Model* com população fixa
Topologia das Ilhas	Totalmente conectada (*Fully Connected*)	Anel e Malha
Critério de Migração	Indivíduo aleatório da ilha com melhor *fitness* médio	Seleção por Torneio
Tipo de Migração	Síncrona	Assíncrona com 15% de chance de ocorrer por geração
Taxa de Migração	Definida como migração de um indivíduo por evento de migração, sem especificação percentual sobre a população.	Fixa em 5%
Critério de Parada	Limite de tempo máximo (Tmax)	Número fixo de gerações (ex.: 3000)
Problema Abordado	CVRP com dados *benchmark* e dados reais	Problema de Roteamento de Veículos Capacitado (CVRP)
Foco Principal	Avaliar impacto da população dinâmica na qualidade e desempenho	Testar a viabilidade de encontrar boas soluções em tempo hábil via paralelismo multicore e busca local
Plataforma Computacional	Executado em CPU multicore (Intel i5), sem evidência de paralelismo multi-thread ou distribuído.	Paralelismo por *threads* em CPU (memória compartilhada)

Fonte: Elaborada pelo autor

3.3 *Acceleration of Genetic Algorithm on GPU CUDA Platform*

A ideia proposta por Janssen e Liew (2019) consiste na utilização de GPUs Nvidia, por meio da plataforma CUDA, que permite o uso de GPUs para tarefas de propósito geral, com o objetivo de acelerar a execução de Algoritmos Genéticos (GAs), com foco no modelo paralelo baseado em ilhas (*Island Model Genetic Algorithm* — IMGA). A proposta central está em modificar as operações evolutivas do GA para que elas sejam totalmente paralelizáveis e mapeáveis de forma eficiente à arquitetura CUDA, explorando ao máximo o paralelismo massivo oferecido pelas GPUs modernas.

A metodologia apresentada utiliza de forma eficaz os recursos da GPU ao atribuir múltiplas *threads* para cada indivíduo da população, dentro de cada ilha. Essa é a principal

inovação do trabalho, uma vez que abordagens anteriores costumavam alocar apenas uma thread por indivíduo, o que já proporcionava ganhos significativos de desempenho. Ao distribuir um grupo de *threads* (*Thread Group* — TG) para cada indivíduo, o artigo permite que as operações evolutivas, como seleção por torneio, *crossover*, mutação e avaliação de aptidão, sejam executadas em paralelo dentro de cada indivíduo. Esse tipo de abordagem permite que diferentes partes de um individuo sejam manipulados ao mesmo tempo acelerando de forma siguinificativa o tempo de execução.

Os resultados do trabalho demonstram que a modificação do Algoritmo Genético e do Algoritmo Genético em Ilhas para execução em GPUs, utilizando a plataforma CUDA e a alocação de múltiplas threads por indivíduo, proporciona ganhos expressivos de desempenho computacional. Nos experimentos realizados com o problema combinatório N-Queens, a abordagem proposta alcançou acelerações de até 24 vezes em relação a uma implementação sequencial em CPU e até 7,7 vezes quando comparada a uma versão GPU com uma única thread por indivíduo. No contexto do modelo em ilhas, os resultados indicaram speedups adicionais de até 7,3 vezes em relação ao IMGA tradicional em GPU e de até 45,5 vezes em comparação com uma implementação multi-threaded em CPU. Além do ganho em tempo de execução, observou-se também uma melhoria na velocidade de convergência do algoritmo, atribuída ao aumento da pressão seletiva proporcionada pelo uso de torneios paralelos com múltiplos indivíduos. Esses resultados evidenciam que a exploração eficiente do paralelismo em nível de GPU não apenas acelera o algoritmo genético, mas também contribui positivamente para a qualidade e rapidez da convergência das soluções.

Diferentemente do trabalho de Janssen e Liew (2019), que explora a arquitetura CUDA para a execução massivamente paralela de algoritmos genéticos em GPUs, este trabalho adota uma abordagem baseada em *multithreading* na CPU, utilizando *threads* concorrentes em núcleos de processamento convencionais. Além disso, enquanto Janssen e Liew (2019). concentram-se na aceleração do modelo em ilhas aplicado ao problema das N-Rainhas, este estudo tem como foco exclusivo a aplicação do *Island Model Genetic Algorithm* ao problema de roteamento de veículos capacitado (CVRP). Um dos principais diferenciais está na investigação do impacto de diferentes topologias de migração fixas — como anel, malha e aleatória —, aspecto que não foi explorado experimentalmente no trabalho de referência, para mais detalhes comparativos consultar o Quadro 4.

Quadro 4 – Comparação entre o trabalho de Janssen e Liew (2019) e este trabalho

Critério	**Janssen & Liew (2019)**	**Este Trabalho**
Modelo de Paralelismo	Paralelismo massivo com CUDA em GPU	Paralelismo com *threads* (memória compartilhada)
Topologia das Ilhas	Topologia em anel	Fixas: Anel e Malha
Critério de Migração	Migração baseada na substituição de piores indivíduos por melhores de ilhas vizinhas	Seleção por Torneio
Tipo de Migração	Assíncrona	Assíncrona com 15% de chance de ocorrer em cada geração
Taxa de Migração	Fixa (4 indivíduos a cada 100 gerações)	Fixa em 5%
Critério de Parada	Número fixo de gerações (10.000 a 25.000)	Número fixo de gerações
Problema Abordado	Problema das N-Rainhas (combinatório)	Problema de Roteamento de Veículos Capacitado (CVRP)
Foco Principal	Aceleração do IMGA em GPU usando múltiplas *threads* por indivíduo	Testar a viabilidade de encontrar boas soluções em tempo hábil via paralelismo multicore e busca local
Plataforma Computacional	GPU Nvidia GTX1060 (CUDA C/C++)	CPU com execução paralela baseada em *threads*

Fonte: Elaborada pelo autor

4 METODOLOGIA

Nesta seção será discutido a metodologia aplicada neste trabalho.

4.1 Pré-processamento de Dados

Esta etapa compreende a leitura e o processamento dos arquivos contendo as informações necessárias para a definição de cada instância do Problema de Roteamento de Veículos com Capacidade (CVRP), além da construção das estruturas de dados indispensáveis para a resolução do problema, como a matriz de distâncias entre os clientes. Essa matriz pode ser construída utilizando a distância euclidiana, calculada com base nas coordenadas dos clientes, conforme apresentado na seguinte equação:

$$d(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{4.1}$$

É importante destacar que o tempo de execução associado ao pré-processamento não será considerado na avaliação de desempenho dos experimentos realizados, tanto para o Algoritmo Genético (AG) sequencial quanto para os diferentes modelos de Algoritmos Genéticos Paralelos (AGPs).

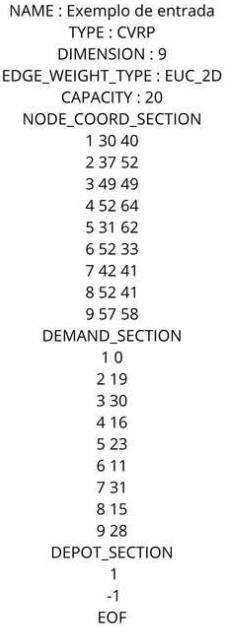
Os dados utilizados neste trabalho serão provenientes do site <http://vrp.galgos.inf.puc-rio.br/index.php/en/>, cuja estrutura padrão da maioria dos arquivos encontra-se representada na Figura 11. A escolha desta base de dados justifica-se por sua ampla aceitação como referência em pesquisas relacionadas ao VRP, além de oferecer *benchmarks* clássicos que fornecem parâmetros confiáveis para a realização de testes comparativos e para a validação dos resultados obtidos.

4.2 Função *fitness*

A função *fitness* é uma parte central da criação e avaliação de soluções adequadas. Sua funcionalidade consiste em avaliar quantitativamente a qualidade de cada indivíduo (solução) da população.

Neste trabalho, a função *fitness* será projetada para refletir o custo total de uma solução do Problema de Roteamento de Veículos com Capacidade (CVRP), considerando a

Figura 11 – Exemplo de entrada CVRP.



Fonte: Adaptada de <https://galgos.inf.puc-rio.br/cvrplib/en/instances>

soma máxima da rota realizada por cada veículo e respeitando o limite máximo de carga de cada veículo.

Sua equação pode ser representada da seguinte maneira:

$$Fitness = \sum_{k=1}^{m} \sum_{i=1}^{n_k-1} d(v_i^k, v_{i+1}^k) \tag{4.2}$$

Onde:

- m representa o número total de veículos disponíveis;
- n_k indica o número de clientes atendidos pelo veículo k;
- $d(v_i^k, v_{i+1}^k)$ corresponde à distância entre o cliente i e o cliente $i+1$ na rota do veículo k.

Soluções que violarem a capacidade máxima de carga dos veículos serão consideradas inviáveis e, caso necessário, poderão ser penalizadas, com a aplicação de um valor sobre o seu valor de aptidão, ou poderão ser corrigidas a fim de torná-las viáveis, assim garantindo que todas as rotas respeitem as restrições do problema.

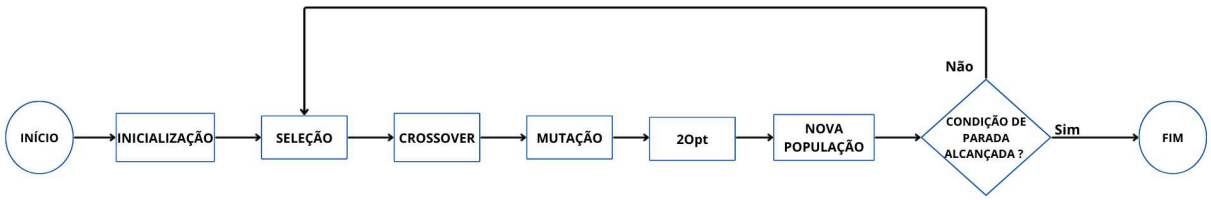
4.3 Algoritmo Memético Sequencial

Nesta etapa, descreve-se a implementação do algoritmo sequencial que servirá como linha de base (baseline) para a análise de qualidade das soluções. Diferente de um

Algoritmo Genético (AG) clássico puro, esta implementação incorpora operadores de busca local, classificando-o como um Algoritmo Memético Sequencial.

A estrutura geral segue o fluxo evolutivo ilustrado na Figura 2, acrescido da etapa de refinamento local. Inicialmente, é gerada uma população de soluções válidas para o CVRP de forma aleatória ou semi-gulosa. A codificação adotada é a representação por permutação de inteiros, onde cada gene representa um cliente. O conjunto de rotas é inferido por um processo de decodificação (split procedure) que respeita a restrição de capacidade dos veículos. A função de aptidão (fitness) é calculada como o inverso da distância total percorrida, visando minimizar o custo logístico.

Figura 12 – Diagrama GA sequencial.



Fonte: Adaptada de Lambora *et al.* (2019).

Para os operadores genéticos, baseando-se em testes preliminares:

- Seleção: Adotou-se o método de Torneio (com tamanho $k = 5$), por oferecer um bom equilíbrio entre pressão seletiva e diversidade, superando a Roleta em estabilidade.
- Cruzamento (Crossover): Utilizou-se o PMX (Partially Mapped Crossover).
- Mutação: Aplicou-se a mutação por troca (*Swap Mutation*) e Inversão de rotas, com uma taxa fixa de 25%.

A aplicação da Busca Local vem imediatamente após a mutação, aplica-se o operador 2-opt aos descendentes gerados. Quanto à parametrização, para garantir uma comparação justa com o modelo paralelo em ilhas, o algoritmo sequencial foi configurado para realizar um esforço computacional equivalente. O critério de parada foi definido em 3.000 gerações, com uma população total de 2.000 indivíduos (equivalente à soma das subpopulações das ilhas nos testes paralelos).

4.4 Algoritmo Genético Paralelo

Para a implementação paralela, adotou-se o Modelo em Ilhas (Island Model), classificado como um algoritmo paralelo de granularidade grossa. A implementação foi realizada utilizando a biblioteca OpenMP em arquitetura de memória compartilhada, onde cada ilha é executada por uma thread independente (núcleo lógico) do processador. A população total (P_{total}) foi fragmentada em N subpopulações (ilhas), de modo que $P_{ilha} = P_{total}/N$, Cada ilha possui seus próprios geradores de números aleatórios, inicializados com sementes (seeds) únicas, os quais são utilizados tanto na geração de novos indivíduos quanto no cálculo de probabilidades ao longo da execução do algoritmo. Essa divisão visa manter o esforço computacional equivalente ao do modelo sequencial, garantindo a justiça na comparação de desempenho. Nos experimentos finais, utilizou-se diferentes quantidades de ilhas, (3, 4, 8 e 12 ilhas) A escolha de iniciar com três ilhas decorre do fato de existirem três configurações distintas de ilhas. Com o objetivo de contemplar todas elas nos testes iniciais, decidiu-se iniciar os experimentos a partir desse valor, para observar como essa mundança afeta o algoritmo de forma geral, sendo o tamamho da população global de 2.000 indivíduos.

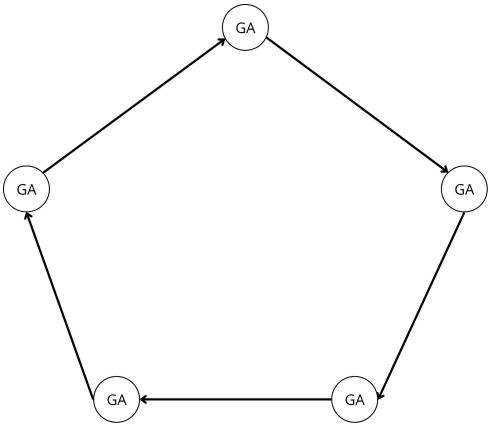
Diferentemente da proposta inicial homogênea, este trabalho implementou um conceito de Ilhas Heterogêneas para maximizar a diversidade. As ilhas foram configuradas com estratégias distintas baseadas em seu identificador (thread_id), alternando parâmetros de mutação e crossover:

- Ilhas de Exploração: Configuradas com altas taxas de mutação, 25% e 50%, para evitar ótimos locais.
- Ilhas de Refinamento: Configuradas com busca local intensiva (2-opt) e taxas de mutação conservadoras, 15%.

Para avaliar o impacto da conectividade, duas topologias foram implementadas e comparadas:

- **Anel (*Ring*)**: As subpopulações são alocadas em uma estrutura circular e a comunicação ocorre exclusivamente entre os vizinhos mais próximos, seguindo o sentido horário (Tomassini, 2005).

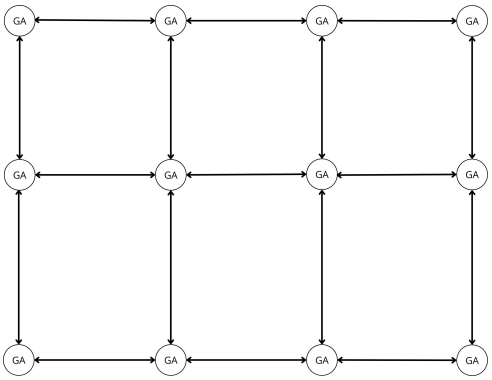
Figura 13 – Exemplo de topologia em anel.



Fonte: Adaptada de Tomassini (2005).

- **Malha**: Ou ‘malhas de ilhas’ são estruturas formadas pela conexão de cada ilha com seus vizinhos mais próximos, formando assim uma estrutura bidimensional bastante conectada (Tomassini, 2005).

Figura 14 – Exemplo de topologia em malha.



Fonte: Adaptada de Alba e Troya (1999).

A comunicação entre as ilhas, realizada por meio da migração, constitui um componente crítico do sistema. Foi implementado um mecanismo de migração assíncrona, no qual, a cada geração, há uma probabilidade de 15% de ocorrer migração. A seleção dos indivíduos migrantes é feita por meio de um torneio, com o objetivo de promover maior diversidade entre as ilhas. Os indivíduos selecionados substituem os piores indivíduos das ilhas receptoras.

4.5 Desenho Experimental

Nesta seeção será descrito o plano experimental para comparar o desempenho do algoritmo genético sequencial (*AG*) com o algoritmo genético paralelo (*AGP*) e suas variações, principalmente no que se refere às topologias.

4.5.1 Objetivos dos Experimentos

- Comparar o desempenho entre a abordagem **Memética Sequencial** e o modelo **Memético Paralelo em Ilhas**, garantindo a equivalência de esforço computacional (mesma população total) para validar os ganhos reais de tempo e qualidade (Gap).
- Avaliar o impacto das topologias de migração (**Anel e Malha**) na disseminação de boas soluções e na manutenção da diversidade genética global, verificando qual estrutura favorece melhor a convergência para o ótimo global.
- Quantificar os ganhos de desempenho através de métricas clássicas de computação paralela, especificamente *Speedup* e Eficiência, analisando a escalabilidade do algoritmo em arquitetura *multicore*.
- Investigar a eficácia da estratégia de **Ilhas Heterogêneas**, analisando se a diversificação de parâmetros (taxas de mutação) e operadores entre as ilhas contribui para evitar a convergência prematura em instâncias complexas.
- Analisar a contribuição do operador de busca local (2-opt) na redução do erro relativo (Gap) e na capacidade de refinamento das rotas, validando a hibridização do algoritmo.

4.5.2 Métricas de Avaliação

- **Qualidade da solução**: avaliada a partir do valor da função *fitness*, na qual valores menores indicam soluções de melhor qualidade. Essa métrica é utilizada como critério principal de avaliação, em consonância com os objetivos deste trabalho.
- **Tempo de execução**: medido em segundos, corresponde ao tempo total necessário para a execução do algoritmo até o critério de parada estabelecido.
- **Métricas de paralelismo** (aplicáveis apenas ao Algoritmo Genético Paralelo): utilizadas de forma complementar para caracterizar o comportamento computacional da abordagem proposta, conforme discutido na Seção 2.5.2 da Fundamentação Teórica, como o speedup e a eficiência.

4.5.3 Configuração dos Experimentos

O Quadro 5 demonstra os parâmetros básicos que pretende-se aplicar aos experimentos que se seguem.

Quadro 5 – Configuração para experimentos

Critério	**Decisão**
Modelo de Paralelismo	Paralelismo via *multithreading* em CPU (núcleos físicos/lógicos)
Topologia das Ilhas	Anel e Malha
Critério de Migração	Seleção por torneio
Tipo de Migração	Assícrona 15% de chance de ocorrer por geração
Taxa de Migração	Fixa em 5%
Critério de Parada	Número fixo de gerações (3000)
Método de Seleção	Torneio (implementação convencional)
Operadores de Crossover	PMX e RBX
Operador de Mutação	*Swap* e Inversão de rotas
Linguagem	C++

4.6 Ambiente Computacional

Os testes apresentados neste trabalho serão realizados em uma máquina com as seguintes configurações: processador AMD Ryzen 5 5600G com 6 nucleos e 12 threads, com tecnologia SMT (Simultaneous Multithreading), 16 GB de memória RAM e sistema operacional Windows 10 Pro. A linguagem de programação escolhida foi C++ devido à sua alta eficiência, juntamente com a API *OpenMP*, que permite a implementação de programação paralela em memória compartilhada.

5 RESULTADOS

Nesta seção são apresentados e analisados os resultados obtidos a partir da execução de três abordagens: o modelo evolutivo sem a utilização do modelo em ilhas, bem como as versões sequencial e paralela do modelo em ilhas. Na abordagem sequencial, as ilhas são processadas de forma intercalada, de modo que apenas uma ilha é evoluída por vez. Já na versão paralela, as ilhas evoluem simultaneamente, permitindo a realização de um maior número de operações evolutivas e avaliações da função fitness dentro do mesmo intervalo de tempo. Dessa forma, embora ambas as abordagens compartilhem a mesma estrutura lógica do modelo em ilhas, a versão paralela apresenta maior capacidade de exploração do espaço de busca sob a mesma restrição temporal. Além dos resultados apresentados a seguir, informações adicionais podem ser encontradas nos Apêndices A, B e C.

5.1 Instâncias trabalhadas

Nesta seção são apresentadas as instâncias utilizadas nos experimentos realizados ao longo deste trabalho. As instâncias representam diferentes configurações do problema estudado e foram selecionadas com o objetivo de avaliar o desempenho e o comportamento do método proposto em cenários variados.

Para cada instância, são descritas suas principais características, tais como tamanho do problema e parâmetros relevantes, de forma a facilitar a compreensão dos resultados obtidos e possibilitar a reprodutibilidade dos experimentos.

Essas instâncias foram divididas em três grupos: as de pequeno porte, com dimensão inferior a 40, apresentadas na Tabela 1.

Tabela 1 – Instâncias de pequeno porte utilizadas

Instância	**Dimensão**	**Caminhões**	**Capacidade**	**Ótimo**
A-n37-k5	37	5	100	669
B-n38-k6	38	6	100	805
E-n33-k4	33	4	8000	835

Fonte: Elaborado pelo autor pelo autor

O segundo grupo é composto por instâncias consideradas de médio porte, com dimensão entre 40 e 99, apresentadas na Tabela 2.

Tabela 2 – Instâncias de médio porte utilizadas

Instância	Dimensão	Caminhões	Capacidade	Ótimo
A-n80-k10	80	10	100	1763
E-n76-k7	76	7	220	682
P-n65-k10	65	10	130	792

Fonte: Elaborado pelo autor pelo autor

Por último, são apresentadas as instâncias consideradas de grande porte, com dimensão superior a 99, conforme a Tabela 3.

Tabela 3 – Instâncias de grande porte utilizadas

Instância	Dimensão	Caminhões	Capacidade	Ótimo
M-n151-k12	151	12	200	1053
M-n200-k17	200	17	200	1373
X-n181-k23	181	23	8	25569

Fonte: Elaborado pelo autor pelo autor

5.2 Análise da Qualidade da Solução

$$Gap(\%) = \frac{S_{obtida} - S_{BKS}}{S_{BKS}} \times 100 \tag{5.1}$$

Onde S_{obtida} é o custo da rota encontrada pelo algoritmo e S_{BKS} é o valor ótimo (ou melhor conhecido) para a instância testada. Um Gap de 0,0% indica que o algoritmo encontrou a solução ótima. Para garantir uma comparação justa, tanto o modelo sequencial quanto o paralelo foram submetidos ao mesmo esforço computacional (mesmo número total de avaliações de função objetivo). As análises a seguir buscam validar a hipótese de que a topologia em ilhas, aliada à busca local, favorece a diversidade genética e permite escapar de ótimos locais onde a abordagem sequencial tende a estagnar.

$$Reduo\ do\ Gap\ (\%) = \frac{Gap_{seq} - Gap_{ilhas}}{Gap_{seq}} \times 100 \tag{5.2}$$

Onde Gap_{seq} representa o valor do Gap obtido pelo algoritmo que não utiliza o modelo em ilhas (sequencial), e Gap_{ilhas} corresponde ao Gap alcançado pelo algoritmo que emprega a abordagem paralela baseada em ilhas.

A redução do Gap indica o quanto a utilização do modelo em ilhas contribuiu para a melhoria da qualidade da solução em relação à abordagem sequencial. Valores positivos de redução do Gap evidenciam que o modelo em ilhas foi capaz de encontrar soluções mais próximas do valor ótimo (ou melhor conhecido), reforçando a eficácia da paralelização na exploração do espaço de busca.

5.2.1 Instâncias de Pequeno Porte

A Tabela 4 apresenta os resultados obtidos para instâncias de pequeno porte, comparando diferentes configurações do modelo em ilhas em relação à abordagem sequencial. Observa-se que, para esse conjunto de instâncias, os valores de *gap* são, em geral, baixos, o que indica que o algoritmo sequencial já é capaz de encontrar soluções próximas do ótimo. Ainda assim, o uso do modelo em ilhas proporciona melhorias consistentes em diversos cenários, especialmente à medida que o número de ilhas aumenta.

De forma geral, configurações com maior número de ilhas tendem a apresentar reduções mais expressivas do *gap*, evidenciando o efeito positivo da diversidade promovida pelo modelo em ilhas, mesmo em instâncias de menor complexidade. Em particular, observa-se que topologias em anel e malha apresentam comportamentos semelhantes, embora a topologia em anel apresente, em alguns casos, reduções de *gap* ligeiramente superiores, como verificado para as instâncias B-n38-k6 e A-n37-k5 com 12 ilhas.

Tabela 4 – Resultados comparativos para instâncias de pequeno porte

Instância	Nº de Ilhas	Topologia	Gap alcançado	Redução do Gap
B-n38-k6	1	—	0,50%	—
B-n38-k6	3	Malha	0,51%	-2,50%
B-n38-k6	4	Malha	0,54%	-7,63%
B-n38-k6	8	Malha	0,46%	14,99%
B-n38-k6	12	Malha	0,42%	14,99%
B-n38-k6	3	Anel	0,52%	-4,99%
B-n38-k6	4	Anel	0,51%	-2,50%
B-n38-k6	8	Anel	0,40%	18,90%
B-n38-k6	12	Anel	0,35%	30,65%
E-n33-k4	1	—	0,35%	—
E-n33-k4	3	Malha	0,24%	31,03%
E-n33-k4	4	Malha	0,38%	-10,35%
E-n33-k4	8	Malha	0,31%	10,34%
E-n33-k4	12	Malha	0,28%	20,69%
E-n33-k4	3	Anel	0,31%	10,34%
E-n33-k4	4	Anel	0,24%	31,03%
E-n33-k4	8	Anel	0,31%	10,34%
E-n33-k4	12	Anel	0,28%	20,70%
A-n37-k5	1	—	1,26%	—
A-n37-k5	3	Malha	0,66%	47,61%
A-n37-k5	4	Malha	0,84%	33,36%
A-n37-k5	8	Malha	0,61%	51,19%
A-n37-k5	12	Malha	0,51%	59,55%
A-n37-k5	3	Anel	0,78%	38,11%
A-n37-k5	4	Anel	1,02%	19,08%
A-n37-k5	8	Anel	0,82%	34,93%
A-n37-k5	12	Anel	0,74%	41,08%

Fonte: Elaborado pelo autor pelo autor

Por outro lado, observa-se que determinadas configurações resultam em reduções negativas do *gap*, indicando que, para instâncias de menor porte, o aumento do número de ilhas nem sempre está associado a melhorias na qualidade da solução. Esse comportamento pode estar relacionado ao baixo custo computacional dessas instâncias, nas quais o algoritmo sequencial tende a convergir rapidamente, o que pode reduzir o impacto do paralelismo e da migração entre ilhas. Esses resultados sugerem que os benefícios do modelo em ilhas tendem a se manifestar de

forma mais evidente em instâncias de maior porte.

5.2.2 Instâncias de Médio Porte

Os resultados obtidos para as instâncias de médio porte indicam que o aumento do número de ilhas contribui para a melhoria da qualidade das soluções encontradas, apesar de demostrar uma certa instabilidade com número menores de ilhas podendo, em certos casos, mesmo que um quantidade considerevalemnte pequena, piorar os resultados atigindo pelo modelo convencional memético (Modelo de uma única ilha com uma população maior).

Tabela 5 – Resultados comparativos para as instâncias E-n76-k7, P-n65-k10 e A-n80-k10

Instância	**Nº de Ilhas**	**Topologia**	**Gap alcançado**	**Redução do Gap**
E-n76-k7	1	—	4,24%	—
E-n76-k7	3	Malha	4,62%	-8,96%
E-n76-k7	4	Malha	3,86%	-8,96%
E-n76-k7	8	Malha	3,96%	7,31%
E-n76-k7	12	Malha	3,93%	7,31%
E-n76-k7	3	Anel	4,11%	3,07%
E-n76-k7	4	Anel	4,63%	-9,20%
E-n76-k7	8	Anel	3,06%	27,83%
E-n76-k7	12	Anel	3,31%	21,97%
P-n65-k10	1	—	3,47%	—
P-n65-k10	3	Malha	2,59%	23,36%
P-n65-k10	4	Malha	3,40%	2,02%
P-n65-k10	8	Malha	2,98%	14,12%
P-n65-k10	12	Malha	2,89%	16,71%
P-n65-k10	3	Anel	2,63%	24,21%
P-n65-k10	4	Anel	3,06%	11,82%
P-n65-k10	8	Anel	3,16%	8,93%
P-n65-k10	12	Anel	2,71%	21,90%
A-n80-k10	1	—	2,27%	—
A-n80-k10	3	Malha	2,37%	-4,50%
A-n80-k10	4	Malha	2,39%	-5,51%
A-n80-k10	8	Malha	2,48%	-9,48%
A-n80-k10	12	Malha	2,11%	6,74%
A-n80-k10	3	Anel	2,28%	-0,48%
A-n80-k10	4	Anel	2,52%	-11,06%
A-n80-k10	8	Anel	2,44%	-7,54%
A-n80-k10	12	Anel	1,85%	18,47%

Fonte: Elaborado pelo autor pelo autor

Em ambas as topologias, observa-se redução do *gap* médio com o aumento do número de ilhas, sobretudo a partir de 8 ilhas, indicando maior diversidade genética e menor convergência prematura. O operador 2-opt contribui adicionalmente para a intensificação da busca, potencializando os resultados do modelo em ilhas.

5.2.3 Instâncias de Grande Porte

Para as instâncias de grande porte, os benefícios do modelo de ilhas tornam-se ainda mais evidentes. Os resultados mostram reduções expressivas do Gap médio com o aumento do número de ilhas, indicando que a abordagem paralela é particularmente adequada para problemas de maior complexidade.

Tabela 6 – Resultados comparativos para as instâncias M-n200-k17, M-n151-k12 e X-n181-k23

Instância	Nº de Ilhas	Topologia	Gap alcançado	Redução do Gap
M-n200-k17	1	—	3,62%	—
M-n200-k17	3	Malha	2,84%	21,47%
M-n200-k17	4	Malha	3,15%	12,93%
M-n200-k17	8	Malha	2,30%	36,53%
M-n200-k17	12	Malha	1,91%	47,27%
M-n200-k17	3	Anel	3,15%	13,09%
M-n200-k17	4	Anel	2,96%	18,31%
M-n200-k17	8	Anel	2,04%	43,67%
M-n200-k17	12	Anel	1,86%	48,70%
M-n151-k12	1	—	3,85%	—
M-n151-k12	3	Malha	4,16%	-8,15%
M-n151-k12	4	Malha	3,76%	2,22%
M-n151-k12	8	Malha	3,21%	16,54%
M-n151-k12	12	Malha	2,95%	23,44%
M-n151-k12	3	Anel	3,94%	-2,46%
M-n151-k12	4	Anel	3,41%	11,37%
M-n151-k12	8	Anel	3,22%	16,30%
M-n151-k12	12	Anel	2,58%	32,84%
X-n181-k23	1	—	2,21%	—
X-n181-k23	3	Malha	1,63%	26,22%
X-n181-k23	4	Malha	1,64%	25,76%
X-n181-k23	8	Malha	1,46%	34,25%
X-n181-k23	12	Malha	1,26%	43,15%
X-n181-k23	3	Anel	1,49%	32,81%
X-n181-k23	4	Anel	1,35%	38,80%
X-n181-k23	8	Anel	1,27%	42,57%
X-n181-k23	12	Anel	1,26%	42,98%

Fonte: Elaborado pelo autor pelo autor

5.3 Análise de Desempenho Computacional

A análise de desempenho computacional foi realizada considerando uma instância representativa de grande porte (M-n200-k17), de modo a avaliar o comportamento do algoritmo em um cenário de maior custo computacional. Para essa análise, foi considerada apenas a topologia de migração em Malha, uma vez que as diferenças de tempo de execução entre as topologias não impactaram significativamente o comportamento paralelo observado.

5.3.1 Tempo de Execução

A Tabela 7 apresenta o tempo médio de 10 execuções por teste do algoritmo para a instância de maior porte considerada (M-n200-k17), considerando tanto o modelo sequencial quanto o modelo em ilhas executado de forma paralela. No modelo sequencial, as ilhas não são executadas simultaneamente, de modo que o tempo total de execução permanece praticamente constante, independentemente do número de ilhas configuradas, sendo equivalente ao tempo observado para a execução com uma única ilha. Em contrapartida, no modelo paralelo, observa-se uma redução significativa do tempo médio de execução à medida que o número de ilhas aumenta, evidenciando que os ganhos de desempenho estão diretamente associados à execução concorrente das ilhas. Ainda assim, tais ganhos não ocorrem de forma linear, indicando a presença de sobrecargas inerentes à paralelização.

Tabela 7 – Tempo médio de execução sequencial e paralelo para a instância M-n200-k17

Número de Ilhas	**Tempo Sequencial (s)**	**Tempo Paralelo (s)**
1	585,80	585,80
3	585,80	316,65
4	585,80	237,75
8	585,80	163,10
12	585,80	135,70

Fonte: Elaborado pelo autor

Observa-se uma redução expressiva do tempo médio de execução à medida que o número de ilhas aumenta, evidenciando a efetividade da paralelização aplicada ao algoritmo. No entanto, os ganhos obtidos não crescem de forma linear.

5.3.2 Speedup

O speedup do algoritmo foi calculado com base no tempo médio de execução da versão sequencial e das versões paralelas, e já explicada na sub-seção 2.5.2.1, onde T_1 representa o tempo de execução da versão sequencial e T_p o tempo obtido com p ilhas.

A Figura 15 apresenta o speedup em função do número de ilhas para a instância de maior porte considerada (M-n200-k17).

Figura 15 – Speedup em função do número de ilhas para a instância M-n200-k17

Fonte: Elaborado pelo autor

Observa-se que o *speedup* aumenta progressivamente com o número de ilhas, sugerindo que a paralelização do algoritmo contribui para a redução do tempo de execução. Entretanto, o comportamento observado é sublinear, o que é comumente reportado em algoritmos meméticos paralelos. Esse resultado pode estar relacionado ao uso exclusivo de memória compartilhada, sem a adoção de abordagens complementares de paralelismo, como ambientes distribuídos ou o uso de aceleradores, por exemplo, GPUs.

5.3.3 Eficiência

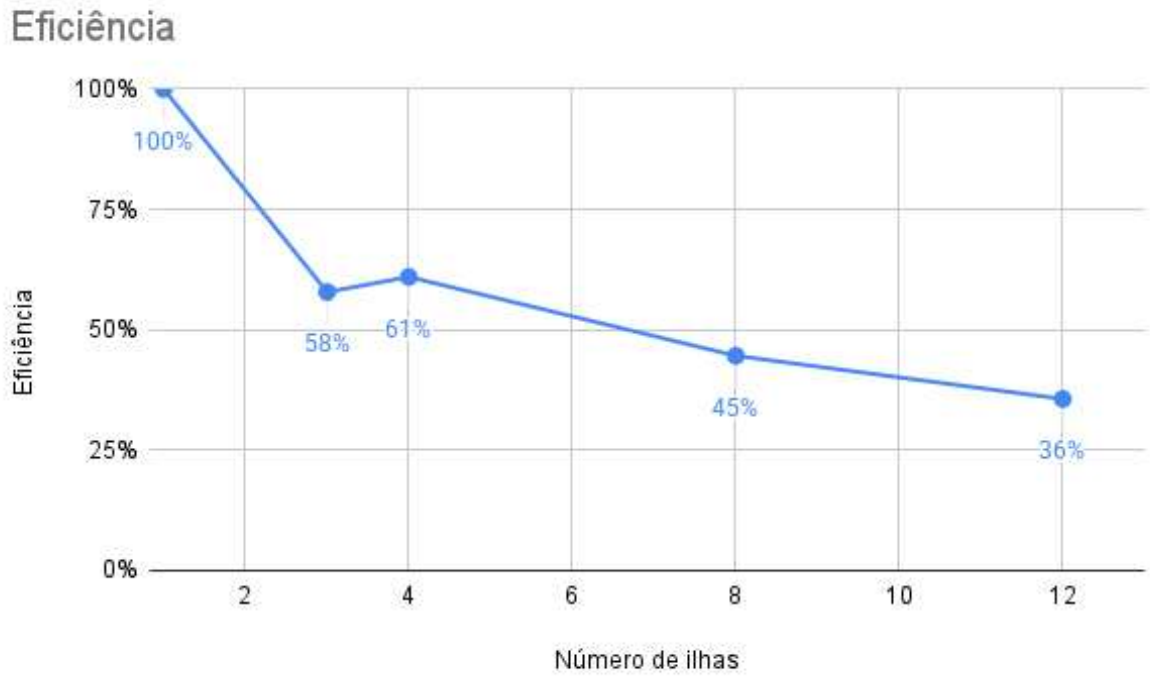
A eficiência paralela do algoritmo foi avaliada conforme a Equação 5.3.3:

$$\text{Eficiência} = \frac{\text{Speedup}}{p}$$

onde p representa o número de ilhas utilizadas na execução paralela.

A Figura 16 apresenta a eficiência paralela em função do número de ilhas para a instância M-n200-k17.

Figura 16 – Eficiência paralela em função do número de ilhas para a instância M-n200-k17



Fonte: Elaborado pelo autor

Observa-se que a eficiência paralela decresce à medida que o número de ilhas aumenta, comportamento esperado em algoritmos paralelos que incorporam mecanismos de comunicação, sincronização e busca local. Apesar dessa redução, os resultados obtidos são coerentes com a proposta do trabalho, uma vez que o paralelismo adotado tem como principal objetivo viabilizar estratégias evolutivas mais robustas e melhorar a qualidade das soluções, sendo o desempenho computacional analisado de forma complementar.

5.4 Análise das topologias

Esta seção apresenta uma análise comparativa entre as topologias de migração em Anel e em Malha no contexto do Algoritmo Memético Paralelo, com o objetivo de avaliar seu impacto na qualidade das soluções obtidas. A análise considera tanto instâncias de grande porte quanto instâncias de porte médio, nas quais os efeitos relacionados à diversidade populacional e à intensificação da busca tendem a se manifestar de maneira distinta.

A Tabela 8 apresenta um resumo comparativo dos valores de Gap médio (%) obtidos para diferentes instâncias, considerando apenas as configurações com 8 e 12 ilhas, que representam cenários com maior grau de paralelismo. Esses resultados permitem uma avaliação mais direta do comportamento das topologias sob condições semelhantes.

Tabela 8 – Comparação do Gap médio (%) entre as topologias de migração para 8 e 12 ilhas

Instância	**Ilhas**	**Anel**	**Malha**	**Melhor**
M-n200-k17	8	2,039	2,298	Anel
M-n200-k17	12	1,857	1,909	Anel
M-n151-k12	8	3,219	3,210	Malha
M-n151-k12	12	2,583	2,945	Anel
X-n181-k23	8	1,271	1,455	Anel
X-n181-k23	12	1,262	1,258	Malha
E-n76-k7	8	3,060	3,960	Anel
E-n76-k7	12	3,310	3,930	Anel
P-n65-k10	8	3,160	2,980	Malha
P-n65-k10	12	2,710	2,890	Anel
A-n80-k10	8	2,440	2,484	Anel
A-n80-k10	12	1,850	2,116	Anel

Fonte: Elaborado pelo autor

De modo geral, observa-se que ambas as topologias são capazes de promover reduções significativas no Gap médio em relação à execução sequencial, confirmando a efetividade do modelo de ilhas. A topologia em Anel apresenta, em diversos cenários, valores de Gap ligeiramente inferiores, especialmente para configurações com 12 ilhas, o que sugere uma melhor preservação da diversidade populacional ao longo da execução.

Por outro lado, a topologia em Malha também demonstra desempenho competitivo, apresentando resultados próximos — e em alguns casos superiores — aos da topologia em Anel, particularmente em instâncias específicas e para determinadas configurações. Sua maior conectividade favorece uma disseminação mais rápida de indivíduos de alta qualidade, o que

pode acelerar a convergência do algoritmo.

Dessa forma, embora os resultados indiquem uma tendência de desempenho ligeiramente superior da topologia em Anel no que se refere à qualidade final das soluções, os dados obtidos não são suficientes para afirmar de maneira conclusiva a superioridade de uma topologia sobre a outra. Esses achados reforçam que a escolha da topologia de migração envolve um compromisso entre intensificação e diversificação da busca, devendo considerar as características da instância e os objetivos da abordagem adotada.

5.5 Avaliação da Qualidade da Solução sob Tempo Fixo

Com o objetivo de analisar o impacto do paralelismo na qualidade das soluções obtidas, foi realizado um experimento adicional considerando um tempo de execução fixo de 200 segundos. Nesse experimento, tanto a versão sequencial quanto a versão paralela do algoritmo utilizam o modelo em ilhas, mantendo as mesmas configurações de operadores genéticos, política de migração, topologia de comunicação e número de ilhas. A diferença entre as abordagens reside exclusivamente na forma de execução das ilhas, sendo sequencial em um único fluxo de execução ou paralela por meio de múltiplas *threads*.

Na abordagem sequencial, as ilhas são processadas de forma intercalada, de modo que apenas uma ilha é evoluída por vez. Já na versão paralela, as ilhas evoluem simultaneamente, permitindo a realização de um maior número de operações evolutivas e avaliações da função *fitness* dentro do mesmo intervalo de tempo. Dessa forma, embora ambas as abordagens compartilhem a mesma estrutura lógica do modelo em ilhas, a versão paralela possui maior capacidade de exploração do espaço de busca sob a mesma restrição temporal.

Os experimentos foram conduzidos utilizando instâncias de maior porte do CVRP, com o objetivo de evidenciar o comportamento do algoritmo em cenários mais desafiadores. A Tabela 9 apresenta o *gap* médio obtido após 200 segundos de execução para diferentes configurações de número de ilhas, considerando a topologia em anel, tanto para a execução sequencial quanto para a paralela.

Tabela 9 – Gap médio obtido após 200 segundos para diferentes instâncias considerando o modelo em ilhas sequencial e paralelo

Nº de Ilhas	**Instância**	**Topologia**	**Forma**	**Gap Médio (%)**
8	X-n181-k23	Anel	Sequencial	1.49
12	X-n181-k23	Anel	Sequencial	1.52
8	X-n181-k23	Anel	Paralelo	1.22
12	X-n181-k23	Anel	Paralelo	1.21
8	M-n151-k12	Anel	Sequencial	4.24
12	M-n151-k12	Anel	Sequencial	3.57
8	M-n151-k12	Anel	Paralelo	3.11
12	M-n151-k12	Anel	Paralelo	2.31
8	M-n200-k17	Anel	Sequencial	3.67
12	M-n200-k17	Anel	Sequencial	3.82
8	M-n200-k17	Anel	Paralelo	1.83
12	M-n200-k17	Anel	Paralelo	1.32

Fonte: Elaborado pelo autor

Como apresentado na Tabela 9, o modelo em ilhas paralelo mostrou-se mais eficiente do que a versão sequencial, como esperado. Isso se deve ao paralelismo, que permite a realização de um maior número de execuções no mesmo intervalo de tempo, possibilitando a obtenção de soluções com melhores valores de GAP de forma mais rápida. Esse comportamento é evidenciado na instância M-n200-k17, na qual o GAP foi reduzido de 3,82% para 1,32%.

6 CONSIDERAÇÕES FINAIS

Este trabalho investigou a aplicação de Algoritmos Genéticos Meméticos Paralelos baseados no modelo em ilhas para a resolução do Problema de Roteamento de Veículos com Capacidade (CVRP), com foco na análise do impacto do paralelismo e da migração entre subpopulações na qualidade das soluções obtidas. Diferentemente de abordagens voltadas exclusivamente ao desempenho computacional, a proposta concentrou-se na avaliação do paralelismo como estratégia algorítmica para promover diversidade genética e intensificar a exploração do espaço de busca.

Os resultados obtidos indicam que o modelo em ilhas contribui positivamente para a melhoria da qualidade das soluções, sobretudo em instâncias de médio e grande porte. Observou-se que o aumento do número de ilhas tende a reduzir o *gap* médio, especialmente a partir de configurações com 8 ilhas, evidenciando que a evolução simultânea de múltiplas subpopulações favorece a mitigação da convergência prematura. Em instâncias de pequeno porte, por outro lado, os ganhos foram mais modestos e, em alguns casos, inexistentes, comportamento esperado devido ao baixo custo computacional e à rápida convergência do algoritmo sequencial.

A análise comparativa entre as topologias de migração em Anel e em Malha demonstrou que ambas são eficazes na promoção de melhorias na qualidade das soluções. Embora a topologia em Anel tenha apresentado, em diversos cenários, valores de *gap* ligeiramente inferiores, os resultados não permitem afirmar de forma conclusiva a superioridade de uma topologia sobre a outra, indicando que a escolha da topologia envolve um compromisso entre diversificação e intensificação da busca, dependente das características da instância e da configuração adotada.

No que se refere ao desempenho computacional, os experimentos evidenciaram reduções significativas no tempo de execução da versão paralela em relação à sequencial, com ganhos de *speedup* sublineares e queda gradual da eficiência à medida que o número de ilhas aumenta, comportamento consistente com a literatura sobre algoritmos meméticos paralelos. Esses resultados confirmam que o paralelismo aplicado viabiliza a execução simultânea das ilhas, ampliando o número de operações evolutivas realizadas por unidade de tempo.

Adicionalmente, o experimento sob tempo fixo reforçou que, mesmo quando ambas as abordagens utilizam o modelo em ilhas, a execução paralela é capaz de obter soluções de melhor qualidade dentro do mesmo orçamento temporal, especialmente em instâncias de maior porte. Esse resultado evidencia que o paralelismo não apenas reduz o tempo de execução, mas também amplia a capacidade de exploração do espaço de busca.

Como proposta para trabalhos futuros, sugere-se a extensão do algoritmo para execução em ambiente de paralelismo em GPU, com o objetivo de explorar ao máximo suas capacidades computacionais. O uso de GPUs, por meio de uma arquitetura mais robusta e eficiente do que o paralelismo em memória compartilhada, pode permitir a realização de um número significativamente maior de operações paralelas. A partir desse ganho computacional, torna-se possível investigar de forma mais aprofundada o impacto do aumento do número de ilhas na qualidade das soluções, analisando até que ponto esse crescimento contribui para a redução do GAP. Dessa forma, busca-se alcançar soluções com GAP igual a zero, ou o mais próximo possível, bem como obter valores de speedup lineares ou próximos do ideal.

REFERÊNCIAS

ALBA, E.; TROYA, J. M. A survey of parallel distributed genetic algorithms. **Complexity**, v. 4, n. 4, p. 31–52, 1999.

BEVILAQUA, A.; BEVILAQUA, D.; YAMANAKA, K. Parallel island based memetic algorithm with lin–kernighan local search for a real-life two-echelon heterogeneous vehicle routing problem based on brazilian wholesale companies. **Applied Soft Computing**, Elsevier, v. 76, p. 697–711, 2019.

CANTÚ-PAZ, E. **A Survey of Parallel Genetic Algorithms**. 1998.

CROES, G. A. A method for solving traveling-salesman problems. **Operations Research**, INFORMS, v. 6, n. 6, p. 791–812, 1958. Disponível em: <https://doi.org/10.1287/opre.6.6.791>.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. **Management Science**, INFORMS, v. 6, n. 1, p. 80–91, 1959.

DAWKINS, R. **The Selfish Gene**. Oxford: Oxford University Press, 1976.

EIBEN, A.; SMITH, J. **Introduction to Evolutionary Computing**. 2nd. ed. Berlin Heidelberg: Springer-Verlag, 2003. (Natural Computing Series).

FLOOD, M. M. The traveling-salesman problem. **Operations Research**, INFORMS, v. 4, n. 1, p. 61–75, 1956. Disponível em: <https://doi.org/10.1287/opre.4.1.61>.

FUNKE, B.; GRÜNERT, T.; IRNICH, S. Local search for vehicle routing and scheduling problems: Review and conceptual integration. **Journal of Heuristics**, Springer Science + Business Media, v. 11, p. 267–306, 2005.

JANSSEN, D. M.; LIEW, A. W.-C. Acceleration of genetic algorithm on gpu cuda platform. In: IEEE. **2019 20th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)**. [S.l.], 2019. p. 211–216.

JOHNSON, D. S.; MCGEOCH, L. A. The traveling salesman problem: A case study in local optimization. **Local Search in Combinatorial Optimization**, Princeton University Press, p. 215–310, 1997.

KUMAR, M.; HUSIAN, M.; UPRETI, N.; GUPTA, D. Genetic algorithm: Review and application. **International Journal of Information Technology and Knowledge Management**, v. 2, n. 2, p. 451–454, 2010. July-December.

LAMBORA, A.; GUPTA, K.; CHOPRA, K. Genetic algorithm- a literature review. In: **2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)**. [S.l.: s.n.], 2019. p. 380–384.

LI, W.; HUANG, Y. A distributed parallel genetic algorithm oriented adaptive migration strategy. In: **2012 8th International Conference on Natural Computation (ICNC)**. [S.l.]: IEEE, 2012. p. 592–596.

MANYIEL, G.; CHEN, T.; GONG, M.; DING, W.; CHAO, F. Multi-population genetic algorithm for rich vehicle routing problems. In: IEEE. **2021 IEEE 7th International Conference on Computer and Information Sciences (ICCOINS)**. [S.l.], 2021. p. 39–44.

MOSCATO, P. **On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms**. Pasadena, CA, 1989.

OHIRA, M.; ISLAM, M. S. Speedup vs. quality: Asynchronous and cluster-based distributed adaptive genetic algorithms for ordered problems. **IEEE Access**, IEEE, v. 8, p. 194744–194760, 2020.

OLUKOTUN, K.; HAMMOND, L. The future of microprocessors. **ACM Queue**, 2005.

POTVIN, J.-Y.; BENGIO, S. The vehicle routing problem with time windows part ii: Genetic search. **INFORMS Journal on Computing**, v. 8, n. 2, p. 165–172, 1996.

RAUBER, T.; RüNGER, G. **Parallel Programming for Multicore and Cluster Systems**. Berlin, Heidelberg: Springer, 2010.

REZAEI, H.; RAJABI, E.; KHODABANDEH, E.; SILVA, D. da; OLIVEIRA, F.; SUBRAMANIAN, M. Exploring dynamic population island genetic algorithm for solving the capacitated vehicle routing problem. **Expert Systems with Applications**, v. 237, p. 122335, 2024. Disponível em: <https://doi.org/10.1016/j.eswa.2024.122335>.

RUSSELL, S. J.; NORVIG, P. **Inteligência Artificial**. Rio de Janeiro: Elsevier, 2013.

SCHMITT, L. M. Fundamental study: Theory of genetic algorithms. **Theoretical Computer Science**, Elsevier, v. 259, n. 1-2, p. 1–61, 2001.

SHAPIRO, J. Genetic algorithms in machine learning. In: PALIOURAS, G.; KARKALETSIS, V.; SPYROPOULOS, C. D. (Ed.). **Machine Learning and Its Applications: Advanced Lectures**. Berlin, Heidelberg: Springer, 2001, (Lecture Notes in Computer Science, v. 2049). p. 146–169.

SUTTER, H. The free lunch is over: A fundamental turn toward concurrency in software. **Dr. Dobb's Journal**, 2005.

TOMASSINI, M. **Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time**. 1. ed. [S.l.]: Springer Berlin, Heidelberg, 2005. (Natural Computing Series).

TOTH, P.; VIGO, D. (Ed.). **The Vehicle Routing Problem**. Philadelphia: SIAM – Society for Industrial and Applied Mathematics, 2002. (Monographs on Discrete Mathematics and Applications).

TOTH, P.; VIGO, D.; IRNICH, S. **Vehicle Routing: Problems, Methods, and Applications**. 2. ed. Philadelphia: SIAM - Society for Industrial and Applied Mathematics, 2014. ISBN 978-1-611974-58-1.

WHITLEY, D.; RANA, S.; HECKENDORN, R. B. Island model genetic algorithms and linearly separable problems. In: **Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)**. Orlando, Florida, USA: Morgan Kaufmann, 1999. p. 110–124.

APÊNDICE A – RESULTADOS COMPLETOS INTÂNCIAS GRANDE PORTE

Tabela 10 – Resultados de desempenho para diferentes números de ilhas e topologias

Nº Ilhas	Instância	Topologia	Gap Médio (%)	Tempo Médio (s)	Speedup	Eficiência	Redução do Gap (%)	Valor Absoluto
1	M-n200-k17	–	3,620	585,80	1,00	100%	–	1.422,70
3	M-n200-k17	Malha	2,843	337,6	1,74	58%	21,47	1.412,03
4	M-n200-k17	Malha	3,152	240,20	2,44	61%	12,93	1.416,27
8	M-n200-k17	Malha	2,298	164,2	3,57	45%	36,53	1.404,55
12	M-n200-k17	Malha	1,909	137,0	4,28	36%	47,27	1.399,21
1	M-n200-k17	–	3,620	585,80	1,00	100%	–	1.422,70
3	M-n200-k17	Anel	3,146	295,7	1,98	66%	13,09	1.416,19
4	M-n200-k17	Anel	2,957	235,3	2,49	62%	18,31	1.413,60
8	M-n200-k17	Anel	2,039	162,0	3,62	45%	43,67	1.401,00
12	M-n200-k17	Anel	1,857	134,4	4,36	36%	48,70	1.398,50
1	M-n151-k12	–	3,846	400,1	1,00	100%	–	1.093,50
3	M-n151-k12	Malha	4,160	172,5	2,32	77%	-8,15	1.096,80
4	M-n151-k12	Malha	3,761	143,9	2,78	70%	2,22	1.092,60
8	M-n151-k12	Malha	3,210	100,7	3,97	50%	16,54	1.086,80
12	M-n151-k12	Malha	2,945	82,8	4,83	40%	23,44	1.084,01
1	M-n151-k12	–	3,846	400,1	1,00	100%	–	1.093,50
3	M-n151-k12	Anel	3,941	178,4	2,24	75%	-2,46	1.094,50
4	M-n151-k12	Anel	3,409	150,3	2,66	67%	11,37	1.088,90
8	M-n151-k12	Anel	3,219	97,8	4,09	51%	16,30	1.086,90
12	M-n151-k12	Anel	2,583	82,1	4,87	41%	32,84	1.080,20
1	X-n181-k23	–	2,213	544,7	1,00	100%	–	26.134,90
3	X-n181-k23	Malha	1,633	274,7	1,98	66%	26,22	25.986,54
4	X-n181-k23	Malha	1,643	216,0	2,52	63%	25,76	25.989,10
8	X-n181-k23	Malha	1,455	161,7	3,37	42%	34,25	25.941,10
12	X-n181-k23	Malha	1,258	131,3	4,15	35%	43,15	25.890,70
1	X-n181-k23	–	2,213	544,7	1,00	100%	–	26.134,90
3	X-n181-k23	Anel	1,487	279,2	1,95	65%	32,81	25.949,21
4	X-n181-k23	Anel	1,354	261,7	2,08	52%	38,80	25.915,31
8	X-n181-k23	Anel	1,271	157,6	3,46	43%	42,57	25.894,01
12	X-n181-k23	Anel	1,262	128,1	4,25	35%	42,98	25.891,67

Fonte: Elaborado pelo autor

APÊNDICE B – RESULTADOS COMPLETOS INSTÂNCIAS MÉDIO PORTE

Tabela 11 – Resultados de desempenho para diferentes números de ilhas e topologias

Nº Ilhas	Instância	Topologia	Gap Médio (%)	Tempo Médio (s)	Speedup	Eficiência	Redução do Gap (%)	Valor Absoluto
1	E-n76-k7	–	4,240	197,60	1,00	100%	–	710,92
3	E-n76-k7	Malha	4,620	104,2	1,90	63%	-8,96	713,51
4	E-n76-k7	Malha	3,860	79,60	2,48	62%	8,96	708,33
8	E-n76-k7	Malha	3,960	60,3	3,28	41%	7,31	709,01
12	E-n76-k7	Malha	3,930	54,9	3,60	30%	7,31	708,80
1	E-n76-k7	–	4,240	197,60	1,00	100%	–	710,92
3	E-n76-k7	Anel	4,110	98,2	2,01	67%	3,07	710,03
4	E-n76-k7	Anel	4,630	80,3	2,46	62%	-9,20	713,58
8	E-n76-k7	Anel	3,060	57,8	3,42	43%	27,83	702,87
12	E-n76-k7	Anel	3,310	56,2	3,52	29%	21,93	704,57
1	P-n65-k10	–	3,470	215,7	1,00	100%	–	819,48
3	P-n65-k10	Malha	2,590	107,0	2,02	67%	25,36	812,51
4	P-n65-k10	Malha	3,400	91,5	2,36	59%	2,02	818,93
8	P-n65-k10	Malha	2,980	58,1	3,71	46%	14,12	815,60
12	P-n65-k10	Malha	2,890	49,6	4,35	36%	16,71	814,89
1	P-n65-k10	–	3,470	215,7	1,00	100%	–	819,48
3	P-n65-k10	Anel	2,630	99,7	2,16	72%	24,21	812,83
4	P-n65-k10	Anel	3,060	86,6	2,49	62%	11,82	816,24
8	P-n65-k10	Anel	3,160	65,5	3,29	41%	8,93	817,03
12	P-n65-k10	Anel	2,710	50,0	4,31	36%	21,90	813,46
1	A-n80-k10	–	2,269	210,30	1,00	100%	–	1.803,00
3	A-n80-k10	Malha	2,371	103,9	2,02	67%	-4,50	1.804,80
4	A-n80-k10	Malha	2,394	90,90	2,31	58%	-5,51	1.805,21
8	A-n80-k10	Malha	2,484	69,3	3,03	38%	-9,48	1.806,79
12	A-n80-k10	Malha	2,116	52,9	3,98	33%	6,74	1.800,31
1	A-n80-k10	–	2,269	210,30	1,00	100%	–	1.803,00
3	A-n80-k10	Anel	2,280	108,4	1,94	65%	-0,48	1.803,20
4	A-n80-k10	Anel	2,520	86,3	2,44	61%	-11,06	1.807,43
8	A-n80-k10	Anel	2,440	57,2	3,68	46%	-7,54	1.806,02
12	A-n80-k10	Anel	1,850	50,1	4,20	35%	18,47	1.795,62

Fonte: Elaborado pelo autor

APÊNDICE C – RESULTADOS COMPLETOS INSTÂNCIAS DE PEQUENO PORTE

Tabela 12 – Resultados de desempenho para diferentes números de ilhas e topologias

Nº Ilhas	Instância	Topologia	Gap Médio (%)	Tempo Médio (s)	Speedup	Eficiência	Redução do Gap (%)	Valor Absoluto
1	B-n38-k6	–	0,497	95,00	1,00	100%	–	809,00
3	B-n38-k6	Malha	0,509	77,6	1,22	41%	-2,50	809,10
4	B-n38-k6	Malha	0,535	67,80	1,40	35%	-7,63	809,31
8	B-n38-k6	Malha	0,460	38,9	2,44	31%	14,99	808,70
12	B-n38-k6	Malha	0,422	33,8	2,81	23%	14,99	808,40
1	B-n38-k6	–	0,497	95,00	1,00	100%	–	809,00
3	B-n38-k6	Anel	0,522	81,1	1,17	39%	-4,99	809,20
4	B-n38-k6	Anel	0,509	68,7	1,38	35%	-2,50	809,10
8	B-n38-k6	Anel	0,403	43,3	2,19	27%	18,90	808,24
12	B-n38-k6	Anel	0,345	32,1	2,96	25%	30,65	807,77
1	E-n33-k4	–	0,347	109,1	1,00	100%	–	837,90
3	E-n33-k4	Malha	0,240	71,9	1,52	51%	31,03	837,00
4	E-n33-k4	Malha	0,383	52,9	2,06	52%	-10,35	838,20
8	E-n33-k4	Malha	0,311	35,8	3,05	38%	10,34	837,60
12	E-n33-k4	Malha	0,275	27,2	4,01	33%	20,69	837,30
1	E-n33-k4	–	0,347	109,1	1,00	100%	–	837,90
3	E-n33-k4	Anel	0,311	71,2	1,53	51%	10,34	837,60
4	E-n33-k4	Anel	0,240	56,0	1,95	49%	31,03	837,00
8	E-n33-k4	Anel	0,311	35,4	3,08	39%	10,34	837,60
12	E-n33-k4	Anel	0,275	26,2	4,16	35%	20,70	837,30
1	A-n37-k5	–	1,256	100,20	1,00	100%	–	677,40
3	A-n37-k5	Malha	0,658	62,1	1,61	54%	47,61	673,40
4	A-n37-k5	Malha	0,837	52,60	1,90	48%	33,36	674,60
8	A-n37-k5	Malha	0,613	40,0	2,51	31%	51,19	673,10
12	A-n37-k5	Malha	0,508	32,0	3,13	26%	59,55	672,40
1	A-n37-k5	–	1,256	100,20	1,00	100%	–	677,40
3	A-n37-k5	Anel	0,777	67,9	1,48	49%	38,11	674,20
4	A-n37-k5	Anel	1,016	63,9	1,57	39%	19,08	675,80
8	A-n37-k5	Anel	0,817	56,77	1,77	22%	34,93	674,47
12	A-n37-k5	Anel	0,740	50,28	1,99	17%	41,08	673,95

Fonte: Elaborado pelo autor