

Universidade Federal do Maranhão  
Centro de Ciências Exatas e Tecnologia  
Curso de Ciência da Computação

**RAMON COSTA SILVA**

ESTUDO SOBRE TOPOLOGIAS DE ALGORITMOS  
GENÉTICOS PARALELOS COM APLICAÇÃO EM  
MINIMIZAÇÃO DE FUNÇÕES NUMÉRICAS IRRESTRITAS

São Luís  
2018

**RAMON COSTA SILVA**

**ESTUDO SOBRE TOPOLOGIAS DE ALGORITMOS  
GENÉTICOS PARALELOS COM APLICAÇÃO EM  
MINIMIZAÇÃO DE FUNÇÕES NUMÉRICAS IRRESTRITAS**

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Alexandre César Muniz de Oliveira

São Luís

2018

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).  
Núcleo Integrado de Bibliotecas/UFMA

Silva, Ramon Costa.

Estudo sobre topologias de algoritmos genéticos paralelos com aplicação em minimização de funções numéricas irrestritas / Ramon Costa Silva. - 2018.  
62 f.

Orientador(a): Alexandre César Muniz de Oliveira.  
Monografia (Graduação) - Curso de Ciência da Computação, Universidade Federal do Maranhão, Sala de Reuniões do NTI - UFMA, 2018.

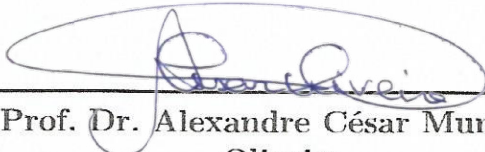
1. Algoritmos Evolutivos Paralelos. 2. Otimização Numérica. 3. Topologias. I. Oliveira, Alexandre César Muniz de. II. Título.

Ramon Costa Silva

# ESTUDO SOBRE TOPOLOGIAS DE ALGORITMOS GENÉTICOS PARALELOS COM APLICAÇÃO EM MINIMIZAÇÃO DE FUNÇÕES NUMÉRICAS IRRESTRITAS

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

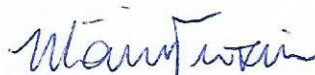
Trabalho aprovado. São Luís, 11 de Julho de 2018.



Prof. Dr. Alexandre César Muniz de  
Oliveira  
(Orientador)  
Universidade Federal do Maranhão



Prof. Msc. Francisco Glaubos Nunes  
Clímaco  
Universidade Federal do Maranhão



Prof. Dr. Mário Antonio Meireles  
Teixeira  
Universidade Federal do Maranhão

São Luís  
2018

*Para Nayara, Sarah, Lourival e Raimunda,  
com todo amor e carinho.*

# Agradecimentos

Agradeço primeiramente a Deus por todas as oportunidades que me deu e tem dado ao longo da caminhada até chegar aqui.

Agradeço a meus pais pelo sustento e compreensão, e a meus irmãos por estarem em todos os momentos desta árdua caminhada, lhes devo tudo o que sou.

Agradeço à minha esposa Nayara, minha melhor amiga e amor de toda a vida, pelo amor, dedicação e pelo suporte em todos os momentos. Agradeço à minha linda filha Sarah por ser um raio de luz e alegria na minha vida. Esta conquista e todas as que virão são todas dedicadas a você, minha filha.

Agradeço ao meu professor e orientador, Alexandre César Muniz de Oliveira pelo ensino, paciência, perseverança e atenção para comigo. Dedico e atribuo grande parte desta vitória você.

Agradeço aos meus amigos das lutas da universidade, do LACMOR e aos amigos de todos estes anos de vida que Deus me tem dado. Em especial a Paulo Jansen, Marcelo Branco, Raphael Gomes, Igor Cavalcanti, Whesley Dantas, Alex Newman, Antônio Mourão e Paulo Edson, pois sempre estiveram dispostos a ajudar, animar e incentivar nesta caminhada. A todos estes meu muito obrigado.

*“Porque dEle, e por Ele,  
e para Ele são todas as coisas;  
glória, pois, a Ele eternamente. Amém!  
(Romanos 11:36)*

# Resumo

Algoritmos Evolutivos sequenciais apesar de efetivos na resolução de problemas de otimização, enfrentam o problema de não conseguirem encontrar soluções viáveis em tempo hábil quando o espaço de busca se torna demasiadamente grande. Buscando solucionar este problema surgem os Algoritmos Evolutivos Paralelos, que são abordagens contemporâneas usadas satisfatoriamente na resolução de instâncias de grande porte de problemas de otimização, pois podem tirar proveito de hardwares sofisticados de alto desempenho, permitindo redução de tempo de execução proporcional ao número de processadores com significativo aumento de eficiência. Fazendo uso do poder destes na resolução de problemas de otimização de grande porte, este trabalho tem por finalidade apresentar um estudo comparativo entre quatro diferentes topologias de Algoritmos Evolutivos Paralelos, aplicadas à minimização de funções numéricas irrestritas. As topologias utilizadas neste trabalho são: Algoritmo Paralelo e Adaptativo com Competição Justa (APHAC), *Ring* e Ilhas. Os experimentos, para este estudo, foram realizados utilizando a biblioteca *Message Passing Interface* (MPI) e a linguagem de programação C no supercomputador disponível no Centro Nacional de Processamento de Alto Desempenho da Universidade Federal do Ceará (CENAPAD-UFC), que é composto por vários servidores em *blade*, podendo atingir um processamento teórico na casa de  $5,363e+03$  GFlops. Com os resultados obtidos, foi possível perceber que o algoritmo proposto utilizando a topologia Ilhas obteve vitória na maioria das análises e, também, utilizar duas subpopulações evoluindo em paralelo acarretou em melhores resultados.

**Palavras-chaves:** Algoritmos Evolutivos Paralelos. Otimização Numérica. Topologias.



# Abstract

Sequential Evolutionary Algorithms despite its effectiveness on optimization problem solving, face the issue that it is hard to find feasible solutions. In order to solve this problem, Parallel Evolutionary Algorithms arise, which are contemporary approaches used satisfactorily in the resolution of large instances of optimization problems, since they can take advantage of sophisticated high performance hardware, allowing a reduction of execution time proportional to the number of processors with significant increase of efficiency. Using the power of these in solving large optimization problems, this paper aims to present a comparative study among four different topologies of Parallel Evolutionary Algorithm, applied to the minimization of unrestricted numerical functions. The topologies used in this work are: Parallel and Adaptive Algorithm with Fair Competition (APHAC), Ring and Islands. The experiments, for this study, were performed using the Message Passing Interface (MPI) library and the C programming language in the supercomputer available at the National Center for High Performance Processing of the Federal University of Ceará (CENAPAD-UFC). several servers in blade, being able to reach a theoretical processing in the house of  $5.363e+03$  GFlops. With the obtained results, it was possible to perceive that the algorithm proposed using the topology Islands obtained victory in the majority of the analyzes and, also, to use two subpopulations evolving in parallel resulted in better results.

**Keywords:** Parallel Evolutionary Algorithms. Numerical Optimization. Topologies.

# Lista de ilustrações

Figura 1 – Codificações mais comuns de indivíduos. . . . .	18
Figura 2 – Cruzamento de 1 ponto . . . . .	22
Figura 3 – Cruzamento Uniforme . . . . .	23
Figura 4 – Mutação Flip Bit . . . . .	24
Figura 5 – Hibridização em um AE (GROSAN; ABRAHAM, 2007) . . . . .	26
Figura 6 – Topologia Mestre-Escravo . . . . .	28
Figura 7 – Topologia da função Schwefel (DEAP, 2012). . . . .	30
Figura 8 – Topologia da função Rosenbrock (DEAP, 2012). . . . .	31
Figura 9 – Topologia da função Rastrigin (DEAP, 2012). . . . .	31
Figura 10 – Topologia da função Griewank (DEAP, 2012). . . . .	32
Figura 11 – Exemplo de estratificação em hierarquia de classes (OLIVEIRA, 2004) . . . . .	41
Figura 12 – Topologia do modelo HFC (OLIVEIRA, 2004) . . . . .	41
Figura 13 – Topologia de <i>Ring</i> . . . . .	42
Figura 14 – Topologia de Ilhas . . . . .	43
Figura 15 – Resultados para Griewank com 50 variáveis. . . . .	45
Figura 16 – Resultados para Griewank com 500 variáveis. . . . .	46
Figura 17 – Resultados para Rastrigin com 100 variáveis. . . . .	47
Figura 18 – Resultados para Rastrigin com 500 variáveis. . . . .	48
Figura 19 – Resultados para Rosenbrock com 50 variáveis. . . . .	49
Figura 20 – Resultados para Rosenbrock com 500 variáveis. . . . .	50
Figura 21 – Resultados para Schwefel com 50 variáveis. . . . .	51
Figura 22 – Resultados para Schwefel com 500 variáveis. . . . .	53
Figura 23 – Speedup para Griewank com 50 variáveis. . . . .	54
Figura 24 – Speedup para Griewank com 500 variáveis. . . . .	55
Figura 25 – Speedup para Rastrigin com 100 variáveis. . . . .	55
Figura 26 – Speedup para Rastrigin com 500 variáveis. . . . .	56
Figura 27 – Speedup para Rosenbrock com 50 variáveis. . . . .	56
Figura 28 – Speedup para Rosenbrock com 500 variáveis. . . . .	57
Figura 29 – Speedup para Schwefel com 50 variáveis. . . . .	57
Figura 30 – Speedup para Schwefel com 500 variáveis. . . . .	58

# Lista de tabelas

Tabela 1 – Funções numéricas irrestritas . . . . .	30
Tabela 2 – Resultado da comparação par a par em Griewank com 50 variáveis . .	46
Tabela 3 – Resultado da comparação par a par em Griewank com 500 variáveis. .	47
Tabela 4 – Resultado da comparação par a par em Rastrigin com 100 variáveis. . .	48
Tabela 5 – Resultado da comparação par a par em Rastrigin com 500 variáveis . .	48
Tabela 6 – Resultado da comparação par a par em Rosenbrock com 50 variáveis. .	49
Tabela 7 – Resultado da comparação par a par em Rosenbrock com 500 variáveis.	50
Tabela 8 – Resultado da comparação par a par em Schwefel com 50 variáveis. . .	52
Tabela 9 – Resultado da comparação par a par em Schwefel com 500 variáveis. . .	53

# Lista de abreviaturas e siglas

AE	Algoritmo Evolutivo
AEH	Algoritmo Evolutivo Híbrido
AEP	Algoritmo Evolutivo Paralelo
AG	Algoritmo Genético
AGP	Algoritmo Genético Paralelo
HFC	Hierarchical Fair Competition
APHAC	Algoritmo Paralelo Hierárquico e Adaptativo com Competição Justa
MPI	Message Passing Interface
A	APHAC
R	Ring
I	Ilhas

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>14</b>
<b>1.1</b>	<b>Objetivos</b>	<b>15</b>
<b>1.2</b>	<b>Organização do Trabalho</b>	<b>15</b>
<b>2</b>	<b>FUNDAMENTAÇÃO TEÓRICA</b>	<b>16</b>
<b>2.1</b>	<b>Algoritmos Evolutivos</b>	<b>16</b>
2.1.1	Definição	16
2.1.2	Seleção	18
2.1.2.1	Seleção por Roleta	18
2.1.2.2	Seleção por Torneio	19
2.1.2.3	Seleção por Ranking Linear	20
2.1.2.4	Seleção com Pressão Auto-Adaptativa	20
2.1.3	Cruzamento	21
2.1.3.1	Cruzamento de 1 Ponto	22
2.1.3.2	Cruzamento Uniforme	22
2.1.3.3	Cruzamento Blend ( $BLX - \alpha$ )	23
2.1.4	Mutação	24
2.1.4.1	Flip Bit	24
2.1.4.2	Uniforme	24
2.1.4.3	Não-Uniforme	24
<b>2.2</b>	<b>Algoritmos Evolutivos Híbridos</b>	<b>25</b>
2.2.1	Definição	25
<b>2.3</b>	<b>Algoritmos Evolutivos Paralelos</b>	<b>27</b>
2.3.1	Definição	27
2.3.2	Modelos de Topologias Paralelas	27
2.3.2.1	Mestre-Escravo	28
2.3.2.2	Multipopulacional	28
2.3.3	Políticas de Migração	29
<b>2.4</b>	<b>Funções Numéricas Irrestritas</b>	<b>29</b>
2.4.1	Definição	29
<b>3</b>	<b>ALGORITMO PROPOSTO</b>	<b>33</b>
<b>3.1</b>	<b>Algoritmo Evolutivo Híbrido Paralelo</b>	<b>33</b>
3.1.1	Parametrização Geral	34
3.1.2	Pré-evolução	35
3.1.3	Consolidação	35

3.1.4	Evolução Com Justiça . . . . .	36
3.1.5	Evolução Sem Justiça . . . . .	38
3.1.6	Busca Local . . . . .	39
3.1.7	Migração . . . . .	40
3.1.7.1	Hierarchical Fair Competition . . . . .	40
3.1.7.2	Ring . . . . .	42
3.1.7.3	Ilhas . . . . .	43
<b>4</b>	<b>EXPERIMENTOS COMPUTACIONAIS . . . . .</b>	<b>44</b>
<b>4.1</b>	<b>Visão Geral . . . . .</b>	<b>44</b>
<b>4.2</b>	<b>Resultados . . . . .</b>	<b>44</b>
4.2.1	Melhor Configuração de Subpopulação . . . . .	45
4.2.1.1	Griewank com 50 variáveis . . . . .	45
4.2.1.2	Griewank com 500 variáveis . . . . .	46
4.2.1.3	Rastrigin com 100 variáveis . . . . .	47
4.2.1.4	Rastrigin com 500 variáveis . . . . .	48
4.2.1.5	Rosenbrock com 50 variáveis . . . . .	49
4.2.1.6	Rosenbrock com 500 variáveis . . . . .	49
4.2.1.7	Schwefel com 50 variáveis . . . . .	50
4.2.1.8	Schwefel com 500 variáveis . . . . .	52
<b>4.3</b>	<b>Speedup . . . . .</b>	<b>53</b>
4.3.1	Griewank com 50 variáveis . . . . .	54
4.3.2	Griewank com 500 variáveis . . . . .	55
4.3.3	Rastrigin com 100 variáveis . . . . .	55
4.3.4	Rastrigin com 500 variáveis . . . . .	56
4.3.5	Rosenbrock com 50 variáveis . . . . .	56
4.3.6	Rosenbrock com 500 variáveis . . . . .	57
4.3.7	Schwefel com 50 variáveis . . . . .	57
4.3.8	Schwefel com 500 variáveis . . . . .	58
<b>4.4</b>	<b>Discussão . . . . .</b>	<b>58</b>
<b>5</b>	<b>CONCLUSÃO . . . . .</b>	<b>59</b>
<b>5.1</b>	<b>Trabalhos Futuros . . . . .</b>	<b>59</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>61</b>

# 1 Introdução

Problemas de otimização de grande porte são encontrados em aplicações práticas na indústria, logística, administração e das engenharias tendo um grande impacto social e econômico (COUTO; SILVA; BARSANTE, 2015). Solucioná-los, por conta da sua complexidade, pode ser inviável quando se utilizam métodos exatos ou mesmo algoritmos heurísticos sequenciais. A proposta deste trabalho é aplicar técnicas de computação paralela em conjunto com heurísticas eficientes na resolução de problemas de otimização aplicados à funções numéricas irrestritas e com um grande número de dimensões. Estas funções são problemas comumente utilizados na avaliação da qualidade de algoritmos de otimização na busca por soluções ótimas. Principalmente pelo fato das características intrínsecas destas funções retratarem funções objetivo de aplicações reais.

Os Algoritmos Evolutivos são heurísticas de otimização e busca que visam resolver problemas complexos tendo como inspiração os processos evolutivos biológicos encontrados na teoria da evolução de Charles Darwin. A ideia principal dos AEs é criar iterativamente populações de possíveis soluções para determinado problema que são codificadas em indivíduos normalmente representados em estruturas de dados vetoriais. A estes indivíduos aplicam-se operadores genéticos como seleção, reprodução e mutação, que visam melhorar a qualidade das soluções codificadas por estes durante todo o processo evolutivo.

Apesar de serem métodos amplamente utilizados na resolução de problemas de otimização, os AEs encontram dificuldade ao lidar com problemas complexos que possuem espaço de busca muito grande. Visando sanar este problema e colaborar com melhorias nos mecanismos de intensificação de busca dos AEs, surgem os Algoritmos Evolutivos Híbridos (AEHs) que, de acordo com (OLIVEIRA, 2004), podem ser definidos como algoritmos evolutivos que empreguem alguma heurística de busca local específica para o problema a ser resolvido paralelamente ao processo evolutivo, como forma de acelerar a busca. Como exemplo temos o Algoritmo Genético (AG) que, neste caso, fica responsável pelo espaço de busca ao criar soluções que representem regiões promissoras e, em paralelo, uma heurística de busca local fica responsável pela intensificação de busca nestas regiões promissoras.

Ainda que a intensificação de busca nos AEHs sequenciais seja um mecanismo de melhoria de desempenho na busca por soluções viáveis, os mesmos encontram outra barreira. Por serem implementados sequencialmente, à medida que o espaço de busca cresce, os AEs precisam utilizar populações com um número muito grande indivíduos, ocasionando um tempo de espera muito longo para obtenção de boas soluções. As heurísticas de busca local utilizadas pelos AEHs, quando lidam com um número grande de variáveis a otimizar também se tornam lentas, ocasionando uma maior lentidão no processo evolutivo como

todo.

Para contornar o problema da lentidão, são propostos os Algoritmos Evolutivos Paralelos (AEPs) que por utilizar o poder computacional das arquiteturas de computadores paralelos, se tornam uma melhoria dos AEs e AEHs sequenciais. Mas, além de melhoria, eles podem ser entendidos como um novo paradigma na computação evolutiva que é capaz de desempenhar uma busca envolvendo novos operadores evolutivos, novos mecanismos de evolução, etc. Os AEPs podem paralelizar os processos evolutivos dos AEs e AEHs, tornando o tempo de execução muito mais satisfatório, possibilitando a distribuição de tarefas entre processos executando-as simultaneamente.

## 1.1 Objetivos

O objetivo principal deste trabalho é apresentar um estudo sobre o comportamento de Algoritmos Evolutivos Paralelos com subpopulações conectadas por diferentes topologias, no qual inclui-se questões relativas a desempenho em problemas de minimização de funções numéricas sem restrição com um grande número de variáveis.

Este trabalho tem, como objetivo secundário, a formulação de um *framework* unificado para algoritmos evolutivos paralelos, com especialização em 3 topologias e com potencial para acréscimo de outras.

## 1.2 Organização do Trabalho

No Capítulo 2 são apresentados os assuntos que fundamentam este trabalho: Algoritmos Evolutivos; Algoritmos Evolutivos Híbridos; Algoritmos Evolutivos Paralelos; e Funções Numéricas Irrestritas. O Capítulo 3 descreve a metodologia para viabilizar a produção deste trabalho. Neste capítulo, são descritos todos os detalhes das técnicas utilizadas desde a concepção do algoritmo à obtenção dos resultados. No Capítulo 4, os dados experimentais obtidos são disponibilizados e analisados estatisticamente. O Capítulo 5 apresenta a conclusão deste trabalho e mostra, também, direcionamentos para trabalhos futuros.



## 2 Fundamentação Teórica

Este capítulo tem por finalidade a apresentação de conceitos básicos que são utilizados como fundamento para o desenvolvimento deste trabalho. São abordados conceitos de Algoritmos Evolutivos (AE), Algoritmos Evolutivos Híbridos (AEH), Algoritmos Evolutivos Paralelos (AEP) e, também, problemas de minimização de funções numéricas irrestritas com grande número de variáveis.

### 2.1 Algoritmos Evolutivos

#### 2.1.1 Definição

Os Algoritmos Evolutivos (AEs) são heurísticas de busca inspiradas nos processos evolutivos biológicos. Baseados na lei de mutação de Hugo Marie de Vries, seleção natural de Charles Darwin e nas leis de hereditariedade de Gregor Mendel, os AEs promovem uma busca em uma população de indivíduos, visando encontrar o indivíduo mais adaptado ao ambiente (OLIVEIRA, 2004). As principais características dos AGs são:

- a) fácil implementação;
- b) adaptabilidade aos mais diversos tipos de problema;
- c) podem encontrar soluções de boa qualidade e até ótimas;
- d) podem ser combinados com outras heurísticas de busca;
- e) podem otimizar um grande número de parâmetros.

A evolução, no que se refere aos AEs, pode ser descrita como a aplicação de operadores genéticos durante um determinado número de gerações em busca do melhor indivíduo viável que resolva um dado problema de otimização, embora não se possa afirmar que este melhor indivíduo vá ser encontrado. A cada geração é construída uma nova população, onde cada indivíduo representa uma possível solução para o problema. Estes indivíduos, em cooperação com outros advindos de populações de gerações anteriores, são combinados para formarem novos indivíduos. Os novos indivíduos são então avaliados para saber quais farão parte da próxima geração. Entretanto, os AE's possuem algumas deficiências, que segundo (OLIVEIRA, 2004), são:

- a) lentidão em comparação com outros métodos, por trabalharem com uma população de soluções;
- b) dificuldade em definir um mecanismo apropriado de codificação e avaliação de soluções em indivíduos;

- c) dificuldade para ajustar adequadamente os valores de seus parâmetros de desempenho;
- d) mecanismos de intensificação de busca pobres, que ocasionam baixas taxas de convergência.

A ideia básica dos Algoritmos Evolutivos segundo (OLIVEIRA, 2004), é manter uma população de indivíduos que representem as soluções candidatas de um dado problema, onde esta população evolui ao longo do tempo através de um processo de competição, onde os indivíduos mais aptos têm maiores chances de sobreviver e se reproduzir. A evolução e conseqüente adaptação dos indivíduos são dadas pela aplicação de operadores evolutivos, onde os melhores indivíduos são selecionados para reproduzirem-se e gerarem novos indivíduos que estejam mais adaptados ao processo de resolução do problema. No Pseudocódigo 1 são demonstrados os passos de um algoritmo evolutivo:

```
1 t = 0;
2 Gere uma população  $P_0$ ;
3 Avalie a aptidão de cada  $X$  em  $P_0$ ;
4 enquanto Condição faça
5   | t = t + 1;
6   | Selecione  $P_t$  de  $P_{t-1}$ ;
7   | Recombine  $P_t$ ;
8   | Avalie a aptidão de cada  $X$  em  $P_t$ ;
9   |  $X^* = \text{Melhor}(P_t)$ ;
10 fim
```

**Algoritmo 1:** Pseudo-código de um AE (OLIVEIRA, 2004)

Os indivíduos em um AE são os responsáveis por representar as soluções do problema a ser resolvido e, a forma mais comum de representação das soluções através da utilização de estruturas de dados vetoriais, onde o vetor é denominado cromossomo, cada posição do vetor é denominada de gene e a qualidade dos indivíduos da população é denominada de *fitness*. Desta maneira, boa parte dos problemas de otimização podem ter suas possíveis soluções representadas por vetores utilizando codificação real ou binária, apesar de que a forma na qual a codificação do indivíduo ocorrerá pode depender especificamente do problema a ser resolvido.

A codificação binária é utilizada quando o problema tem domínio discreto e pode representar valores numéricos, grafos ou mesmo estruturas de codificação de indivíduos mais complexas. Em contrapartida a codificação binária pode ser um dos principais fatores que decorrem na lentidão do algoritmo, isto acontece quando o número de genes do cromossomo se torna muito grande levando o algoritmo a despender um grande esforço

computacional. Quando o domínio do problema a ser resolvido é contínuo, a codificação mais indicada é a real, pois no espaço de busca contínuo a precisão da solução é um fator que requer muita atenção (CUNHA et al., 2016).



Figura 1 – Codificações mais comuns de indivíduos.

## 2.1.2 Seleção

A principal característica dos Algoritmos Evolutivos, é o fato da busca por soluções viáveis dentro da população ser feita de forma estocástica. O objetivo do operador de seleção é melhorar a qualidade média das soluções da população, atribuindo uma alta probabilidade de seleção a um indivíduo que represente uma boa solução para o problema (BLICKLE, 1995).

De acordo com (BESSAOU; SIARRY, 2001), a seleção nos AEs ocorre quando na população são encontrados dois (ou mais) indivíduos que representem boas soluções e o primeiro passo na busca por estes indivíduos é realizar o cálculo do *fitness* de todos os indivíduos da população. Na literatura existem diversos métodos utilizados para selecionar estes melhores indivíduos. Dentre os mais conhecidos estão a Seleção por roleta, por torneio, por ranking linear e etc. O método de seleção de indivíduos para este trabalho é abordado e melhor explanado no Capítulo 3.

### 2.1.2.1 Seleção por Roleta

A principal característica deste método é que ele provê para cada indivíduo  $i$  da população uma probabilidade  $P(i)$  proporcional ao seu *fitness*,  $f(i)$ . A variável  $n$  significa o tamanho da população em termos de quantidade de indivíduos como demonstrado no pseudocódigo 2, segundo (JEBARI, 2013). A Equação 2.1 modela matematicamente a probabilidade de seleção de indivíduos deste método.

$$P(i) = \frac{f(i)}{\sum_{j=1}^n f(j)} \quad (2.1)$$

```

1 calcula-se a soma  $S \leftarrow \sum_{i=1}^n f(i)$ ;
2 para  $1 \leq i \leq n$  faça
3   |  $\alpha \leftarrow \text{aleatório}(0, S)$ ;
4   |  $soma \leftarrow 0; j \leftarrow 0$ ;
5   | faça
6   |   |  $soma \leftarrow soma + f(i)$ ;
7   |   |  $j \leftarrow j + 1$ ;
8   | enquanto  $soma < \alpha$  e  $j < n$ ;
9   |   | selecione o indivíduo  $j$ ;
10 fim

```

**Algoritmo 2:** Seleção por Roleta (JEBARI, 2013)

### 2.1.2.2 Seleção por Torneio

Neste método a seleção ocorre buscando aleatoriamente um número  $k$  de indivíduos da população e, então, o indivíduo com o melhor *fitness* deste grupo é levado a uma subpopulação intermediária, repetindo este processo várias vezes para toda a população. Normalmente o torneio ocorre entre dois indivíduos, mas pode ser generalizado para um grupo maior de indivíduos. Ao final do torneio o indivíduo com melhor *fitness* é selecionado para o cruzamento. O processo de seleção no método do torneio ocorre como demonstrado no pseudocódigo 3 e a modelagem matemática da probabilidade de seleção, na Equação 2.2.

$$P(i) = \begin{cases} \frac{C_{n-1}^{k-1}}{C_n^k} & \text{se } i \in [1, n - k - 1] \\ 0 & \text{se } i \in [n - k, n] \end{cases} \quad (2.2)$$

```

1 subpop  $\leftarrow n$  indivíduos selecionados para o torneio;
2 para  $1 \leq i \leq n$  faça
3   | para  $1 \leq j \leq n$  faça
4   |   |  $\text{indivíduo}_1 \leftarrow \text{subpop}(i)$ ;
5   |   | para  $1 \leq k \leq n$  faça
6   |   |   |  $\text{indivíduo}_2 \leftarrow \text{subpop}(j + m)$ ;
7   |   |   | se  $f(\text{indivíduo}_1) > f(\text{indivíduo}_2)$  então
8   |   |   |   | selecione  $\text{indivíduo}_1$ ;
9   |   |   | senão
10  |   |   |   | selecione  $\text{indivíduo}_2$ ;
11  |   |   | fim
12  |   | fim
13  |   |  $j = j + k$ ;
14  | fim
15 fim

```

**Algoritmo 3:** Seleção por Torneio (JEBARI, 2013)

### 2.1.2.3 Seleção por Ranking Linear

Este método pode ser entendido como uma variante do método de seleção por roleta, pois foi criado para contornar seu principal defeito – a convergência prematura (SHUKLA; PANDEY; MEHROTRA, 2015). No método de seleção por ranking linear a seleção se dá mais pelo ranking do indivíduo do que pelo *fitness* do mesmo. O primeiro passo do método é ordenar os indivíduos de acordo com seu *fitness*, atribuindo ao pior indivíduo a posição 1 e ao melhor a posição  $N$ . A probabilidade de seleção dos indivíduos é dada de acordo com seus rankings de acordo com as equações 2.3 e 2.4, e o pseudocódigo 11 demonstra computacionalmente o funcionamento do método segundo (BLICKLE, 1995).

$$p_i = \frac{1}{N} \left( n^- + (n^+ - n^-) \frac{i - 1}{N - 1} \right); i \in [1, \dots, N] \quad (2.3)$$

$$j = \lfloor \frac{N}{2(c-1)} \left( c - \sqrt{c^2 - 4(c-1)\chi} \right) \rfloor \quad (2.4)$$

```

1 ranking_linear( $n^-$ ,  $J_1, \dots, J_n$ );
2  $\bar{J} \leftarrow$  população  $J$  ordenada de acordo com o fitness do pior indivíduo;
3  $S_0 \leftarrow 0$ ;
4 para  $1 \leq i \leq N$  faça
5   |  $s_i \leftarrow s_i + p_i$  (Equação 2.3);
6 fim
7 para  $1 \leq i \leq N$  faça
8   |  $r \leftarrow$  aleatório( $0, s_n$ );
9   |  $J'_i \leftarrow \bar{J}_l$  em que  $s_l - 1 \leq r \leq s_l$ ;
10 fim
11 retorne  $J'_1, \dots, J'_N$ 

```

**Algoritmo 4:** Método de Seleção por Ranking Linear (BLICKLE, 1995)

### 2.1.2.4 Seleção com Pressão Auto-Adaptativa

Este método é assim chamado por conta do próprio indivíduo determinar sua pressão de seleção de forma dinâmica a cada seleção. A pressão de seleção do indivíduo tem seu comportamento definido por dois parâmetros: O número  $ns_k$  de vezes que foi selecionado e seu *fitness* com relação ao melhor indivíduo encontrado. Pelo fato dos indivíduos serem selecionados de acordo com um certo limiar de *fitness*, o método SPAA é muito semelhante ao método de seleção por roleta (OLIVEIRA, 2004). A equação demonstra matematicamente como cada indivíduo contribui para o processo de seleção,

onde  $Z(s_k)$  é a avaliação seletiva de cada indivíduo.

$$Z(s_k) = \frac{1}{|f^* - f(s_k) + 1|^{ns_k}} \quad (2.5)$$

Segundo (OLIVEIRA, 2004), o trecho  $|f^* - f(s_k)|$  da equação denota a diferença do indivíduo  $s_k$  e o melhor indivíduo encontrado. Com o intuito de evitar problemas numéricos  $ns_k$  é considerado com um valor máximo, normalmente 5. É importante salientar que à medida em que novos melhores indivíduos são encontrados durante o processo evolutivo, a avaliação seletiva de cada indivíduo muda, mostrando que o método SPAA também é adaptativo.

```

1   $n \leftarrow$  total de indivíduos na população;
2  posição  $\leftarrow$  indivíduo aleatório  $\in [0, n]$ ;
3   $Z \leftarrow$  (Equação 2.5);
4  gatilho  $\leftarrow -Z * (GIRO - (GIRO - 1))$ ;
5  soma  $\leftarrow 0$ ;
6   $i \leftarrow 0$ ;
7  enquanto soma < gatilho e  $i \leq GIRO$  faça
8  |   se  $p < n - 1$  então
9  |   |   posição  $\leftarrow$  posição + 1;
10 |   senão
11 |   |   posição  $\leftarrow 0$ ;
12 |   fim
13 |   se  $ns_k > 5$  então
14 |   |    $ns_k \leftarrow 5$ ;
15 |   fim
16 |    $Z \leftarrow$  (Equação 2.5);
17 |   soma  $\leftarrow Z$ ;
18 |    $i \leftarrow i + 1$ ;
19 fim
20  $ns_k \leftarrow ns_k + 1$ ;
21 retorne posição;

```

**Algoritmo 5:** Seleção Auto-Adaptativa (OLIVEIRA, 2004)

### 2.1.3 Cruzamento

Baseado também na teoria da evolução das espécies de Charles Darwin, o cruzamento é um importante operador evolutivo dos AEs. De acordo com uma certa probabilidade o cruzamento pode acontecer entre dois ou mais indivíduos e ocorre fazendo-se a combinação de partes de cada um destes, atingindo assim o seu principal objetivo que é trocar informações realizando a geração de novos indivíduos a partir dos que foram previamente selecionados pelo método de seleção. Alguns dos principais métodos de cruzamento existentes são: Cruzamento de 1 Ponto, Cruzamento Uniforme, Cruzamento Aritmético, Cruzamento Blend ( $BLX - \alpha$ ) e etc.

### 2.1.3.1 Cruzamento de 1 Ponto

Amplamente usado, o cruzamento de um ponto foi proposto por John Holland em 1975 (HOLLAND, 1975). Este método seleciona um ponto aleatório no vetor de bits dos indivíduos escolhidos no método de seleção. Quando este ponto específico é escolhido, ele faz separação dos indivíduos em duas partes e consecutivamente fazendo a permuta destas partes.

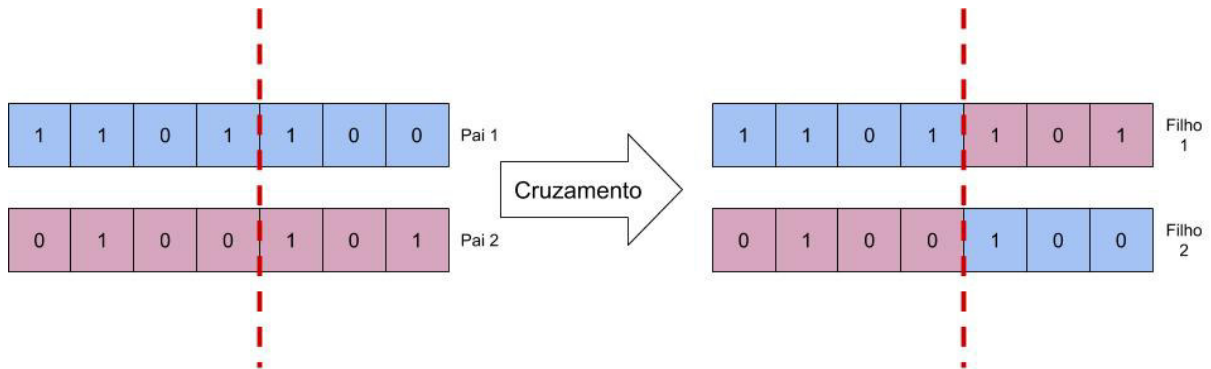


Figura 2 – Cruzamento de 1 ponto

Observando a Figura 2, nota-se que após o cruzamento ser efetuado a troca de informação entre os indivíduos gera dois novos indivíduos, denominados de filhos. Os indivíduos resultantes do cruzamento podem representar soluções de qualidade ou não, dependendo da escolha do local do ponto de corte.

### 2.1.3.2 Cruzamento Uniforme

O Cruzamento Uniforme não define pontos de corte para realizar a recombinação dos genes dos indivíduos. Neste método são definidas máscaras de bits aleatórias para cada par de pais com a mesma quantidade de genes dos mesmos, onde dependendo do bit da máscara o gene específico dos pais deverá ser copiado para o filho ou não, de acordo com a Figura 3.

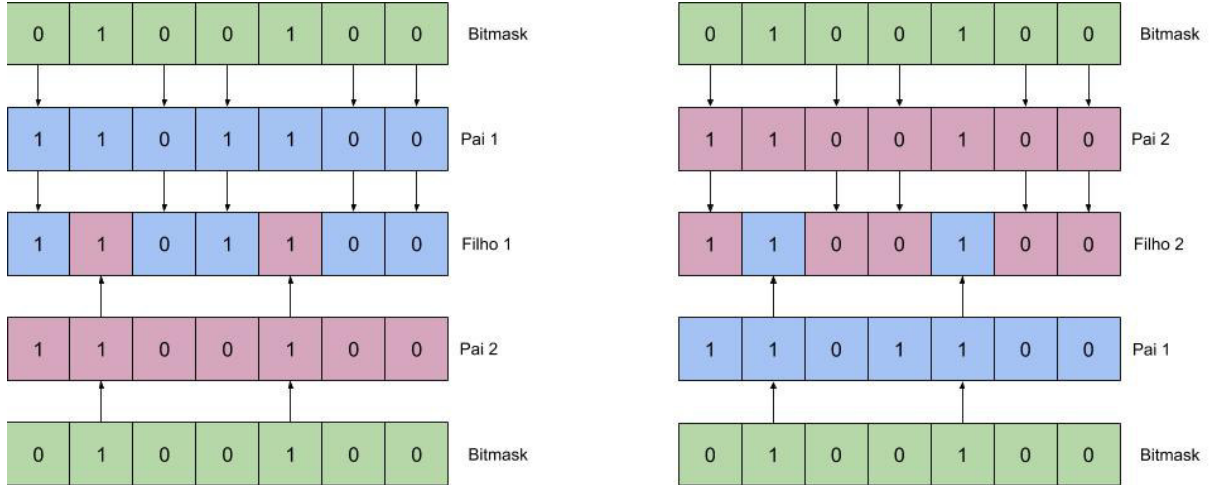


Figura 3 – Cruzamento Uniforme

Como a Figura 3 demonstra, se o bit na máscara for 0 então o gene referente àquela posição no primeiro pai deverá ser copiado para o primeiro filho, se o bit da máscara for 1 então será copiado o respectivo gene do segundo pai para o primeiro filho. Dessa maneira, para gerar o segundo filho basta inverter a ordem dos pais.

### 2.1.3.3 Cruzamento Blend ( $BLX - \alpha$ )

Este método é mais comumente utilizado quando os indivíduos precisam codificar soluções de problemas contínuos. De acordo com (MEHRA M.L. JAYALAL, 2014), para dois pais  $X_t$  e  $X'_t$  na geração  $t$ . Assumindo que  $X_{k,t} < X'_{k,t}$ , se  $X_{k,t}$  e  $X'_{k,t}$  são dois indivíduos a serem submetidos ao cruzamento, então o  $BLX - \alpha$  escolhe uma solução no intervalo  $[X_{k,t} - \alpha(X'_{k,t} - X_{k,t}), X_{k,t} + \alpha(X'_{k,t} - X_{k,t})]$ . Dessa maneira, a Equação 2.6 modela os genes resultantes dos filhos, com o número aleatório  $u \in [0, 1]$ .

$$X_{k,t+1} = (1 - \gamma) * X_{k,t} + \gamma * X'_{k,t} \quad (2.6)$$

Onde  $\gamma = (1 + 2\alpha) * u - \alpha$  e, segundo a Equação 2.7 a localização do indivíduo filho depende da diferença entre os pais.

$$X_{k,t+1} - X_{k,t} = \gamma * (X'_{k,t} - X_{k,t}) \quad (2.7)$$

Se essa diferença for pequena a diferença entre os filhos e os pais também será pequena, o que caracteriza uma busca adaptativa em que nos estágios iniciais a busca ocorrerá por todo espaço de soluções e nos estágios finais a busca será mais concentrada em um local (MEHRA M.L. JAYALAL, 2014).



## 2.1.4 Mutação

A mutação é o operador evolutivo que tem a responsabilidade de não permitir que a busca se concentre em apenas um local do espaço de soluções. Assim como o cruzamento, a mutação também acontece de acordo com uma certa probabilidade. Este operador evolutivo toma o indivíduo gerado após o cruzamento e modifica um gene aleatoriamente, este fato faz com que seja agregada mais diversidade de soluções à população, evitando que o algoritmo caia em ótimos locais prematuramente. Existem muitos tipos de métodos de mutação, dentre eles estão: *Flip Bit*, Uniforme, não-uniforme e etc.

### 2.1.4.1 Flip Bit

Este método é muito usado quando a codificação do indivíduo é discreta. A mutação ocorre simplesmente trocando o valor bit do indivíduo, se o valor for 0, será trocado para 1 e, se for 1 será trocado para 0. A Figura 5 demonstra mais claramente este processo.

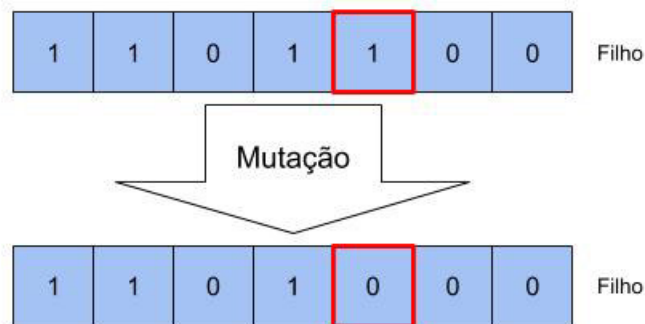


Figura 4 – Mutação Flip Bit

### 2.1.4.2 Uniforme

O método de mutação uniforme é usado quando os indivíduos codificam soluções reais. Este método substitui o gene escolhido por um valor dentro de uma faixa de valores com limites inferior e superior escolhidos previamente.

### 2.1.4.3 Não-Uniforme

Este método visa lidar com o principal problema da aleatoriedade da mutação em indivíduos com codificação real. De acordo com (ZHAO; GAO, 2004) o método é descrito como segue: Para cada indivíduo  $X_i^t$  em uma população na geração  $t$ , criar um indivíduo filho se  $X_i^t = \{x_1, x_2, \dots, x_m\}$  é um cromossomo e o gene  $x_k$  for escolhido para mutação, o

cromossomo resultado  $x'_k$  é obtido através da Equação 2.8.

$$x'_k = \begin{cases} x_k + \Delta(t, UB - x_k) & \text{se um } \xi \text{ aleatório é } 0 \\ x_k - \Delta(t, x_k - LB) & \text{se um } \xi \text{ aleatório é } 1 \end{cases} \quad (2.8)$$

O gene  $x_k$  tem como  $LB$  e  $UB$  por limites inferior e superior respectivamente. A função  $\Delta(t, y)$  retorna um valor no intervalo  $[0, y]$  tal que  $\Delta(t, y)$  tende a zero à medida que  $t$  aumenta. Esta característica faz com que o espaço seja explorado uniformemente nas primeiras gerações e mais localmente ao fim da evolução e, também, faz com que o novo número gerado seja muito mais próximo do seu sucessor do que por uma escolha aleatória.

Onde  $\Delta(t, y) = y(1 - r^{(1 - \frac{t}{T})^b})$ , sendo  $r \in [0, 1]$  um número aleatório uniforme,  $T$  o número máximo de gerações e  $b$  um parâmetro de sistema que determina o grau de dependência do número de gerações.

## 2.2 Algoritmos Evolutivos Híbridos

### 2.2.1 Definição

Muitos trabalhos relacionados aos AEHs vem sendo desenvolvidos no decorrer dos anos utilizando algoritmos baseados em populações para a busca global, e algoritmos que trabalham no melhoramento de soluções para a busca local (LIU, 2014). Desse modo os AEHs se configuram como uma melhoria dos AEs, mas apesar de os EAs tradicionais serem métodos muito bons em resolver problemas de otimização, quando estes problemas se tornam mais e mais complexos, os AEs tradicionais encontram dificuldade em encontrar soluções viáveis em tempo hábil. Neste ponto, a proposta dos AEHs é bem vinda, desde que os mesmos se utilizam de heurísticas de busca local para acelerar o desempenho da busca pelo melhor indivíduo, esta característica dos AEHs é onde reside a principal diferença entre os AEs genéricos. De acordo com (GROSAN; ABRAHAM, 2007), as principais características de um AEH são:

- a) Melhorar a performance dos AEs em termos de velocidade de convergência, por exemplo;
- b) Melhorar a qualidade das soluções obtidas pelos AEs;
- c) Incorporar os AEs como sendo parte de um sistema maior.

OS AEs podem ser facilmente adaptados a qualquer tipo de problema como métodos de otimização global podendo encontrar soluções boas e até ótimas. Mas apesar deste fato os estes esbarram na lentidão da busca quando submetidos a problemas com espaço de busca extensos. Além do problema de lentidão ao lidar com o problema de espaços de busca muito grandes, esta lentidão também ocorre pelo fato de os AEs realizarem a busca

em uma população de soluções candidatas e não trabalhar no melhoramento de uma única solução como ocorrem nas heurísticas de busca local.

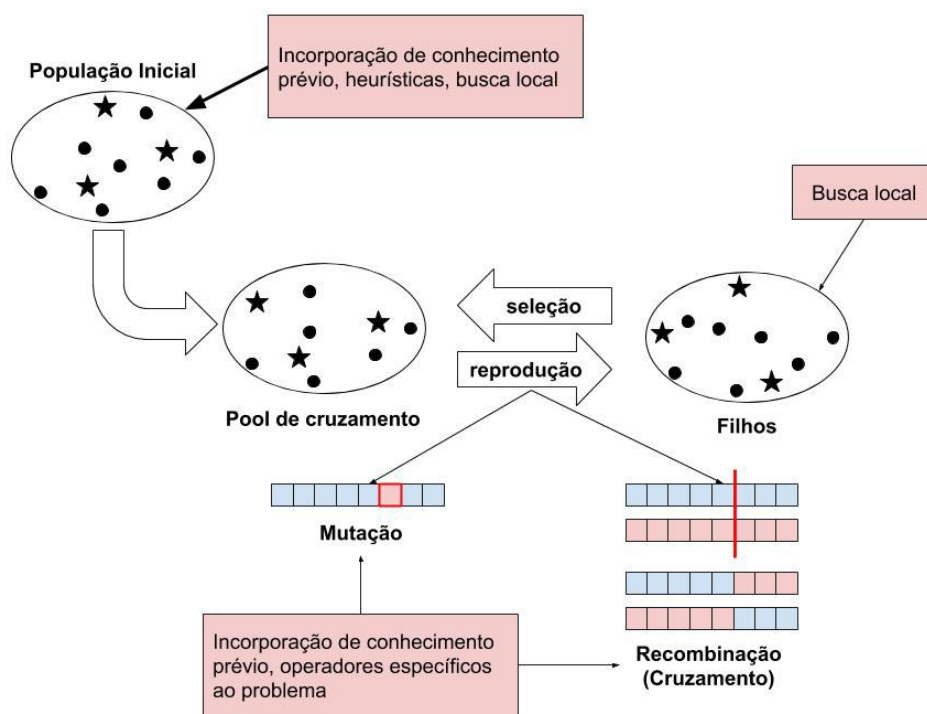


Figura 5 – Hibridização em um AE (GROSAN; ABRAHAM, 2007)

Os AEHs surgem para contornar justamente este problema, pois se mostram técnicas superiores ao utilizar métodos de busca local paralelamente ao processo de evolução para o melhoramento das soluções, com o objetivo de intensificar a busca e, conseqüentemente, melhorar o desempenho do algoritmo. Ainda que estes se utilizem de mecanismos de intensificação de busca para promover uma melhora no desempenho, os AEs encontram um outro obstáculo a superar. Por serem implementados sequencialmente, os AEH's acabam se tornando lentos na busca quando precisam otimizar muitas variáveis.

Quando o problema a ser resolvido tem muitas variáveis a otimizar, as heurísticas de busca local utilizadas por pelos AEs também se tornam lentas, ocasionando uma lentidão ainda maior no processo de evolução e busca como um todo. Esta característica negativa dos AEH's é um problema indesejável na área da otimização. Em um problema com muitas variáveis o espaço de busca se torna extenso, fazendo com que os AEH's necessitem de populações grandes de soluções candidatas para resolução de um problema, tornando impraticável o tempo de espera para obtenção de soluções viáveis. Apesar de os AEH's se utilizarem destes mecanismos visando a melhora do desempenho, quando as heurísticas de busca local tem um custo computacional elevado, aplicá-las indiscriminadamente a todos os indivíduos no processo de evolução torna o processo de busca por soluções de qualidade significativamente lento.

## 2.3 Algoritmos Evolutivos Paralelos

### 2.3.1 Definição

Nos últimos anos nota-se uma crescente popularização e desenvolvimento de arquiteturas computacionais paralelas, desde um simples smartphone que hoje em dia conta com o poder computacional de um processador de vários núcleos, a uma GPU com centenas e milhares de núcleos ou mesmo supercomputadores que possuem capacidade de processamento na casa dos Teraflops. Ligado a este fato, crescem também as pesquisas na área e o desenvolvimento de algoritmos que visam fazer o melhor uso destas poderosas arquiteturas computacionais na resolução de problemas de alta complexidade. Segundo (GRAMA et al., 2003 apud COSTA, 2010) para um bom design e funcionamento de um algoritmo paralelo, as seguintes premissas ou parte delas devem ser seguidas:

- a) Identificar tarefas que podem ser processadas simultaneamente;
- b) Atribuir as tarefas identificadas a diferentes processos;
- c) Distribuir os dados às tarefas.
- d) Gerenciar os dados compartilhados.
- e) Sincronizar os processos.

Os Algoritmos Evolutivos Paralelos são uma melhoria dos AEs e AEHs sequenciais, mas além de melhoria, eles se configuram como um novo paradigma na computação evolutiva que é capaz de desempenhar uma busca envolvendo novos operadores evolutivos, novos mecanismos de evolução, etc (OLIVEIRA, 2004). Os AEs e AEHs sequenciais despendem um tempo computacional muito grande e inviável na procura por soluções em problemas com espaço de busca muito grande, portanto necessita-se de outros meios para contornar esta ineficiência em relação ao tempo de execução.

Para solucionar este problema, são utilizados os AEPs, que por paralelizar os processos evolutivos dos AEs e AEHs, tornam o tempo de execução muito mais satisfatório, tornando possível a distribuição de tarefas entre processos e executando as mesmas simultaneamente. Além do ganho de desempenho computacional em relação ao tempo, os AEPs abrem as portas para a utilização de mecanismos evolutivos de busca mais robustos, como estratificar uma população de acordo com a aptidão de cada indivíduo, atribuindo cada um dos mesmos a uma subpopulação específica, em conformidade ao modelo Hierarchical Fair Competition encontrado em (HU et al., 2002).

### 2.3.2 Modelos de Topologias Paralelas

A paralelização de AEs e AEHs pode ocorrer paralelizando-se os processos evolutivos mais custosos, ou mesmo executando varias instâncias populacionais ao mesmo tempo.

Nos tópicos a seguir, são abordadas algumas das topologias mais utilizadas na literatura.

### 2.3.2.1 Mestre-Escravo

A topologia Mestre-Escravo trabalha com a ideia da paralelização dos processos evolutivos dos AEs e AEHs que mais despendem tempo computacional. Com esta metodologia, pode-se utilizar o processo mestre para manter a população e os processos escravos para cuidar das tarefas como cruzamento, mutação, busca local e etc.

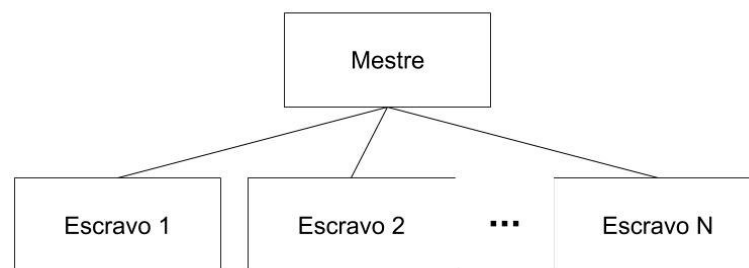


Figura 6 – Topologia Mestre-Escravo

Esta topologia é caracterizada como um modelo de paralelização global por manter uma população de indivíduos globalmente e paralelizar apenas os processos evolutivos. Este tipo de modelo de paralelização não tem a preocupação de saber se a arquitetura em que será implementada é de memória compartilhada ou distribuída, pois se utilizar memória compartilhada a população pode ser mantida na memória compartilhada e cada processo lê seu conjunto de indivíduos, se for implementada em memória distribuída a população pode ficar em um processo mestre que envia os indivíduos para outros os processos realizarem suas tarefas e após o término recolher os resultados (PESSINI, 2003).

### 2.3.2.2 Multipopulacional

Os AEP's com subpopulações que evoluem em paralelo, podem ser classificados dentro do modelo de múltiplas populações ou multipopulacionais. Neste modelo, as subpopulações são distribuídas nas várias unidades de processamento existentes e evoluem isoladamente em paralelo umas às outras. Em conexão com o conceito de subpopulações, surge a ideia de cooperação entre as mesmas. Segundo (OLIVEIRA, 2004), a cooperação entre subpopulações está fundamentada em novos operadores evolutivos, especificamente criados para promover o intercâmbio entre subpopulações baseado no movimento de indivíduos entre as mesmas. A migração é um exemplo de operador específico para a cooperação entre populações.

É válido, também, ressaltar a importância do conceito de granularidade quando se trata da comunicação entre subpopulações nos modelos multipopulacionais, que segundo

(OLIVEIRA, 2004) se trata da razão entre computação e comunicação. A granularidade fina tem por principal característica um custo de comunicação elevado em comparação ao custo de processamento relativo a cálculos e, a granularidade grossa vai no caminho oposto (NOWOSTAWSKI; POLI, 1999 apud OLIVEIRA, 2004).

### 2.3.3 Políticas de Migração

O processo de cooperação entre subpopulações necessita de políticas que possam controlar o movimento dos indivíduos que são enviados de uma subpopulação para outra. Para esta finalidade, são definidas políticas de migração que são estabelecidas de acordo com os seguintes parâmetros, essenciais ao bom funcionamento dos AEPs:

- a) A topologia que define as conexões entre subpopulações;
- b) A frequência com que são realizadas as migrações de indivíduos;
- c) O número de indivíduos que são trocados a cada migração;
- d) O critério de escolha dos imigrantes.

A topologia das conexões entre subpopulações é de suma importância, pois define as restrições de comunicação entre as mesmas em termos de rotas migratórias. A interação entre as subpopulações se dá por meio da comunicação entre as unidades de processamento disponíveis no ambiente paralelo em que o AEP será executado.

## 2.4 Funções Numéricas Irrestritas

### 2.4.1 Definição

Funções numéricas irrestritas, são funções que não possuem restrições na forma de equações  $A(x) = b$  ou inequações  $A(x) \leq b$  (BAZARAA; JARVIS; SHERALI, 1990 apud OLIVEIRA, 2004). Estas são amplamente usadas na avaliação do desempenho de algoritmos de otimização, no que diz respeito a encontrar soluções viáveis em tempo hábil. Funções numéricas irrestritas podem ser unimodais que significa ter um único ponto de ótimo global, ou multimodais que possuem diversos pontos de ótimo local. Estes tipos de funções são problemas particularmente difíceis de resolver por conta da sua característica de possuir muitos pontos de ótimo local. Segundo (CORTES; SILVA, 2010 apud JUNIOR; COSTA; CORTES, 2017), estas funções tem a capacidade de ser escritas como a soma de diversas funções de uma variável, por esse motivo são caracterizadas como separáveis. Entretanto, por conta desta característica, encontrar o ótimo global se torna uma tarefa muito mais difícil pois a solução dependerá de muitas variáveis.

Na literatura existem diversos exemplos de funções numéricas irrestritas sendo utilizadas como medidor da qualidade de algoritmos de otimização. De acordo com

(OLIVEIRA, 2004), para estas funções são previamente definidos os limites inferiores  $x_i^{inf}$  e superiores  $x_i^{sup}$  que as variáveis da função podem assumir sendo parte do problema, de maneira que limitam o espaço de busca sobre regiões mais desafiadoras. A Tabela 1, apresenta o valor ótimo de cada função, dimensionalidade e seus limites inferiores e superiores, assim como as Figuras 7 a 10 apresentam graficamente a topologia de cada uma delas.

Tabela 1 – Funções numéricas irrestritas

Função	Variáveis	Ótimo	$x_i^{inf}; x_i^{sup}$
Schwefel	n	0	-500; 500
Rosenbrock	n	0	-5,12; 5,12
Rastrigin	n	0	-5,12; 5,12
Griewank	n	0	-600; 600

1. Schwefel:

$$f_{sch}(x) = 418,982887n + \sum_{i=1}^n -x_i \sin\left(\sqrt{|x_i|}\right), \quad (-500 \leq x_i \leq 500) \quad (2.9)$$

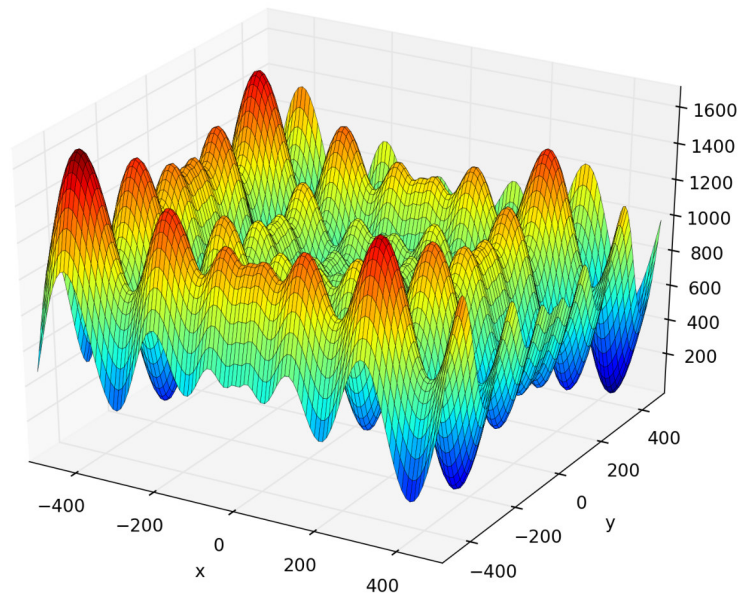


Figura 7 – Topologia da função Schwefel (DEAP, 2012).

2. Rosenbrock:

$$f_{ros}(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2), \quad (-5, 12 \leq x_i \leq 5, 12) \quad (2.10)$$

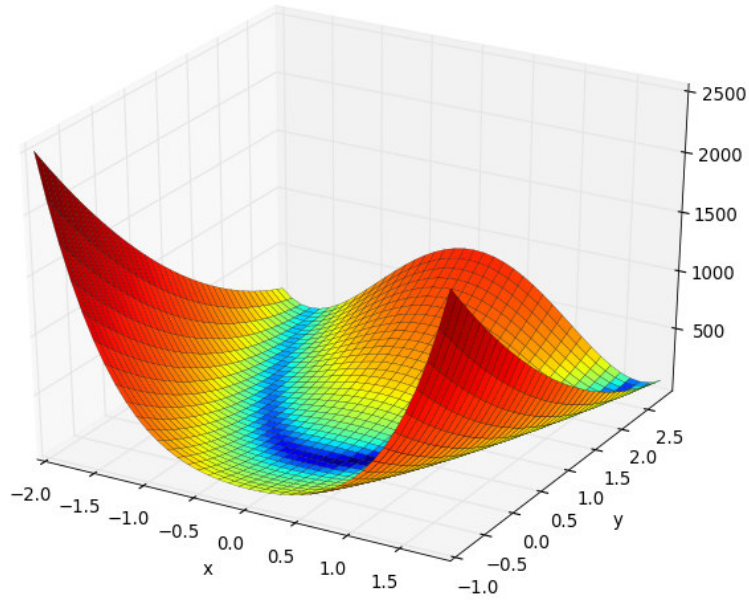


Figura 8 – Topologia da função Rosenbrock (DEAP, 2012).

3. Rastrigin:

$$f_{ras}(x) = 10n + \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i)), \quad (-5, 12 \leq x_i \leq 5, 12) \quad (2.11)$$

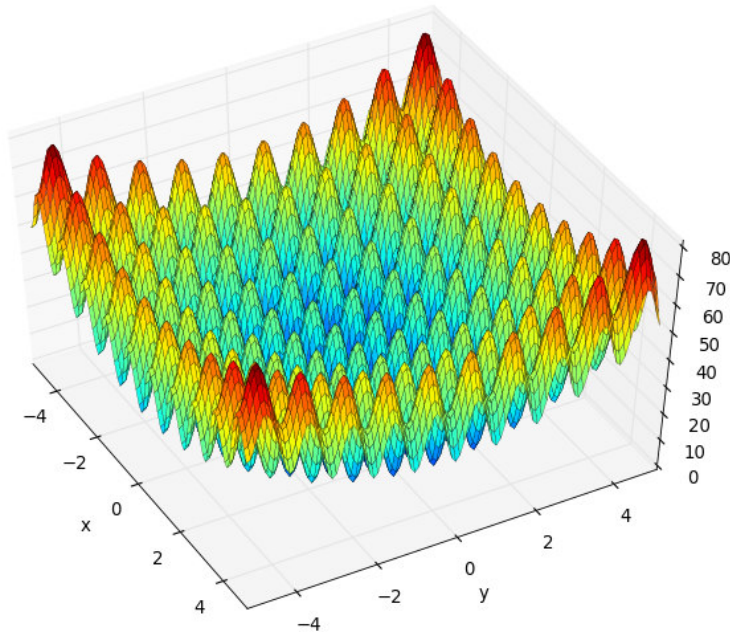


Figura 9 – Topologia da função Rastrigin (DEAP, 2012).

4. Griewank:

$$f_{gri}(x) = 1 + \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \left( \cos\left(\frac{x_i}{\sqrt{i}}\right) \right), \quad (-600 \leq x_i \leq 600) \quad (2.12)$$



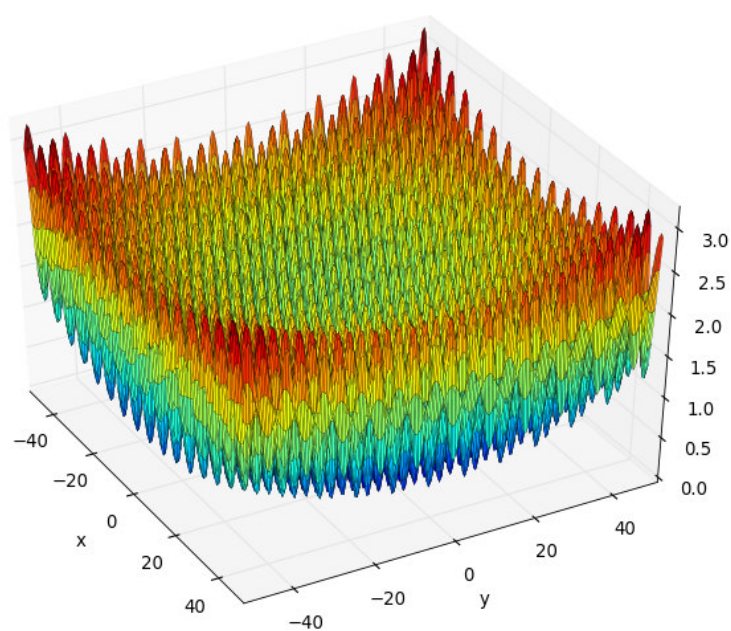


Figura 10 – Topologia da função Griewank (DEAP, 2012).

## 3 Algoritmo Proposto

Este capítulo é dedicado à explicação detalhada de cada passo dado para a concepção do algoritmo que implementa as topologias de algoritmos evolutivos paralelos para posterior análise dos resultados obtidos descritos no Capítulo 4.

### 3.1 Algoritmo Evolutivo Híbrido Paralelo

Para este trabalho, foi utilizado um Algoritmo Genético que é replicado em cada processo do ambiente paralelo, fazendo o papel de agente de busca global na população de soluções para cada problema, enquanto o algoritmo de busca de *Hooke e Jeeves* atua como agente de busca local (HOOKE; JEEVES, 1961). Este algoritmo possui algumas fases principais de execução que estão descritas no pseudocódigo 6.

```

1 Inicializa ambiente paralelo;
2 para  $P_i \in \mathbf{P}$  faça
3   Inicializa( $P_i$ );
4   EvoluiPré( $P_i$ , poucasIterações);
5   Consolida( $fit_{inf}$ ,  $fit_{sup}$ ,  $AT_i$ );
6   faça
7     Critério de superioridade:  $AT_i$ ;
8     se Competição Justa então
9        $\mathbf{Q} = \text{EvoluiJusto}(P_i, \text{máximoIterações}, AT_i)$ ;
10      EnviaIndivíduoSuperior( $\mathbf{Q}$ );
11      RecebeIndivíduosSuperiores( $P_i > 0$ );
12      ComplementaIndivíduosAleatoriamente( $P_0$ );
13     senão
14        $\mathbf{Q} = \text{EvoluInjusto}(P_i, \text{máximoIterações}, AT_i)$ ;
15       EnviaIndivíduoSuperior( $\mathbf{Q}$ );
16       RecebeIndivíduosSuperiores( $P_i$ );
17     fim
18 enquanto critério de parada;
19 fim

```

**Algoritmo 6:** Pseudocódigo do algoritmo proposto

No Algoritmo 6, as linhas de 3 a 5 são responsáveis pela inicialização das subpopulações, evolução em poucas iterações para definição dos parâmetros locais de mapeamento de

faixas de *fitness* e consolidação destes parâmetros a nível global. Na linha 7 são realizados cálculos para definir o critério de superioridade, que caracteriza um indivíduo como superior ou não com o fim de participar do processo de migração.

Após a fase inicial, o algoritmo entra no processo de evolução, podendo ser justa ou injusta dependendo da topologia adotada. Durante o processo de evolução são selecionados indivíduos para um local temporário na memória  $Q$ , para posteriormente serem enviados para determinadas subpopulações e também recebidos de outras subpopulações.

A principal diferença entre as abordagens de competição justa e injusta, é o fato de que na competição justa os indivíduos quando são selecionados para a migração, são movidos para o *buffer* de saída e não participam mais da evolução na subpopulação. Enquanto na competição injusta os indivíduos são apenas copiados para o *buffer* de saída, continuando a fazer parte do processo evolutivo.

### 3.1.1 Parametrização Geral

De acordo com o que foi dito na Seção 3.1, um AG foi utilizado e este foi parametrizado utilizando uma população de 100 indivíduos codificando soluções reais. Foram criadas também duas populações auxiliares de mesmo tamanho denominadas *buffer* de entrada e *buffer* de saída que são responsáveis por armazenar temporariamente os indivíduos que foram selecionados para a fase de migração (envio e recebimento). Foram definidas, também, probabilidades para os operadores evolutivos sendo que apenas dois pais podem ser selecionados a cada geração, 100% de chance de ocorrer cruzamento, no máximo 20% de chance de ocorrer mutação e busca local. No ambiente paralelo que é usado como plataforma para este algoritmo, a biblioteca *Message Passing Interface* (MPI) é responsável por atribuir uma instância deste AG em cada processador do ambiente.

A abordagem de seleção utilizado no algoritmo proposto foi o método de Seleção com Pressão Auto-Adaptativa (SPAA) descrito no Capítulo 2 e o número de pais selecionados a cada iteração da evolução foi fixado em 2. O método SPAA tem uma importante característica para este algoritmo que é a adaptabilidade da seleção a medida que a população muda durante o processo evolutivo.

Para este método, o tipo cruzamento escolhido foi o Cruzamento *Blend* que já foi detalhado na fundamentação teórica no Capítulo 2, e o seu parâmetro  $\alpha$  foi definido em valores que variam entre  $[\alpha_{baixo}, \alpha_{alto}]$  que valem 13% e 33% respectivamente.

A mutação utilizada neste algoritmo é a mutação não-uniforme, esta foi descrita em detalhes no Capítulo 2. A probabilidade de ocorrer a mutação é definida num intervalo de  $[muta_{baixa}, muta_{alta}]$  com valores de 5% a 20%.

Uma importante característica deste algoritmo é que tanto a mutação quanto o parâmetro  $\alpha$  do cruzamento variam dentro dos seus intervalos pré-definidos e, variam

também, de acordo com o ranking de cada subpopulação e estes parâmetros podem ser calculados para cada subpopulação em conformidade com a Equação 3.1:

$$\begin{aligned}
 P_0 &= V_{baixo} \\
 P_{D-1} &= V_{alto} \\
 P_{0 < i < D-1} &= V_{baixo} + R_i \left[ \frac{V_{alto} - V_{baixo}}{1,7 * (D - 2)} \right]
 \end{aligned} \tag{3.1}$$

Onde  $P_i$  é a população,  $D$  é o número total de subpopulações,  $R_i$  é o ranking da população atual e  $V$  é um parâmetro parâmetro que pode ser a probabilidade de mutação ou o valor  $\alpha$  do cruzamento, que variam entre um limite inferior ( $V_{baixo}$ ) e superior ( $V_{alto}$ ) previamente definidos.

### 3.1.2 Pré-evolução

Neste estágio do algoritmo, os parâmetros evolutivos são ajustados previamente de forma que hajam poucas gerações de evolução, servindo como uma preparação para que o processo migração e competição justa possam funcionar de forma correta. Para este passo, o número de gerações foi definido como 100, chance de ocorrer cruzamento com 100% de probabilidade, parâmetro  $\alpha$  do cruzamento em 0,25 e a mutação com probabilidade de 20%. Como a pré-evolução tem o fim apenas de preparar a população para outros estágios, a busca local não tem necessidade de ocorrer.

A pré-evolução é um dos principais estágios deste algoritmo pois evolui cada subpopulação em um determinado número pequeno de gerações, para que possam ser definidos os parâmetros que são responsáveis, no modelo HFC *Hierarchical Fair Competition*, pela definição dos indicadores domínio de cada subpopulação para que o processo de migração possa ocorrer da melhor maneira. Para topologias que não utilizem o modelo HFC, o cálculo destes indicadores domínio não é interessante, já que as rotas de migração são definidas previamente pela topologia empregada.

### 3.1.3 Consolidação

A fase de consolidação é essencial no que tange ao bom funcionamento do modelo HFC, pois neste momento ocorre um importante estágio de comunicação entre todas as subpopulações. Neste ponto todas as subpopulações já tem calculados seus indicadores locais de domínio, e então todos estes dados são enviados para uma determinada subpopulação para consolidação e posterior reenvio para cada subpopulação. Os indicadores locais de domínio de cada subpopulação são submetidos a alguns cálculos na subpopulação elite tornando-se indicadores globais de domínio.

De acordo com (OLIVEIRA, 2004), os indicadores locais de domínio são baseados na média, desvio padrão e o melhor *fitness* e para a definição destes, é necessário o mapeamento de um intervalo para normalização dos valores de *fitness*. Problemas corriqueiros na pré-evolução são encontrar indivíduos fora da faixa de *fitness* referente aos indicadores locais domínio e ocorrer o colapso de faixas de *fitness* que é quando dependendo da topologia da função a ser resolvida, as faixas de *fitness* podem se torna muito estreitas ocasionando problemas numéricos nos cálculos dos limiares de admissão.

A principal etapa da consolidação é o momento da comunicação entre as populações em que os valores dos indicadores locais de domínio de todas as subpopulações são enviadas para a subpopulação elite, que faz cálculos utilizando parâmetros como a menor média de valores *fitness*, menor média de valores *fitness* acima da média, o número total de subpopulações envolvidas no processo, assim como o ranking da subpopulação atual que é definido pelo ambiente criado pela biblioteca MPI.

Todos estes cálculos realizados na subpopulação que realiza a consolidação dos dados enviados pelas outras, são usados para definir os limiares de admissão  $AT_i$  de cada subpopulação de forma que possam ser corrigidos os problemas de colapso de faixas de *fitness* e de indivíduos fora da faixa de valores de *fitness*.

### 3.1.4 Evolução Com Justiça

A evolução com justiça para funcionar corretamente, se utiliza dos *indicadores de domínio* tanto locais quanto globais que foram calculados e consolidados na etapa de consolidação. O processo de evolução nesta abordagem ocorre de forma que todos os indivíduos em todas as subpopulações possam competir com indivíduos que possuam valor de *fitness* equivalente, evitando que indivíduos que representem soluções muito boas sejam sempre selecionados em detrimento dos piores, levando o algoritmo a convergir prematuramente para soluções ótimas locais.

Nesta metodologia ocorre uma estratificação das subpopulações, o que caracteriza a existência de subpopulações em que o processo evolutivo trabalha em cima de indivíduos que representam soluções muito boas e, outras subpopulações que evoluem indivíduos que representam soluções não tão boas mas que ainda assim são importantes para o processo evolutivo como um todo.

Em algoritmos evolutivos, a convergência é desejável mas deve-se ter algum controle sobre ela para evitar os ótimos locais. A evolução com justiça, devido a topologia paralela que utiliza, permite que o risco de convergência prematura seja reduzido de uma forma elegante. O processo de evolução nos algoritmos evolutivos funciona em cima da substituição dos piores indivíduos pelos melhores – o que caracteriza uma certa injustiça – os piores indivíduos que são substituídos podem conter informações genéticas importantes para

encontrar o ótimo global do problema. Deste modo os piores indivíduos são mantidos em subpopulações com limiares de admissão mais baixos e evoluem entre si, preservando estas importantes informações.

Esta metodologia também trabalha com um interessante mecanismo de geração contínua de indivíduos. Por conta da etapa de migração, o espaço onde os indivíduos migrados fica vazio pois estes são movidos para o *buffer* de saída. Dessa maneira foi preciso elaborar este mecanismo de geração de indivíduos para contornar este problema, e o ponto forte deste mecanismo é o fato de ele gerar indivíduos aleatoriamente o que ocasiona no aumento da variedade de soluções. Este processo de geração de indivíduos acontece em apenas uma subpopulação específica e ajuda a diminuir o risco de convergência prematura pois ocorre o aumento na variedade de soluções tanto na população em que ocorre o processo, quanto nas outras subpopulações tendo em mente o fato de que eventualmente estes novos indivíduos aleatórios poderão ser migrados para outras subpopulações.

Neste algoritmo a seleção de indivíduos para a fase de migração, acontece ao mesmo tempo que a seleção para o cruzamento pois os pais escolhidos são avaliados segundo um critério de superioridade e enquanto possíveis pais forem encontrados dentro do critério de superioridade da migração, a fase de seleção não será completada. O critério de superioridade na seleção dos indivíduos diz que um indivíduo é superior se o ranking da subpopulação atual dele, é menor do que o *ranking* da população para a qual ele foi selecionado para migrar e, se o *fitness* dele for diferente do *fitness* do melhor indivíduo encontrado na subpopulação. O critério de superioridade precisa da saber para qual subpopulação o indivíduo selecionado deverá migrar, este dado é obtido na Equação 3.2 que segue:

$$subpop_{rank} = \left\lceil \min(D - 1, \max(0, (\bar{f} - f_{ij}) * \delta)) \right\rceil \quad (3.2)$$

Onde  $D$  é o número total de subpopulações,  $\bar{f}$  é a média de *fitness* da subpopulação,  $f_{ij}$  é o *fitness* do indivíduo selecionado e  $\delta$  é um parâmetro que varia segundo a média  $\bar{f}$  e o desvio padrão  $f_\sigma$  dos valores de *fitness* da subpopulação e é obtido pela Equação 3.3:

$$\delta = \frac{(D - 2)}{(\bar{f} - f_\sigma)} \quad (3.3)$$

Este detalhe na fase de seleção tem um papel importante na abordagem de evolução com justiça, pois se o indivíduo é superior, deve ser migrado para que a competição não se torne injusta na subpopulação em que foi encontrado pois seu *fitness* por ser melhor que os dos demais acaba fazendo com que a probabilidade de ele ser selecionado para o cruzamento seja maior que a dos demais indivíduos na subpopulação. Em termos gerais, este modelo de seleção faz com que o algoritmo apesar de ser diversificado devido o mecanismo de geração contínua de indivíduos e ter o método de seleção SPAA e o Cruzamento *Blend* como

mecanismos para trabalhar esta diversificação, o algoritmo constitui uma característica bastante elitista na busca pelo ótimo global do problema.

Após todo o processo de seleção tanto para o cruzamento quanto para a migração e o próprio cruzamento em si, ocorre a mutação ou a busca local. Neste ponto é importante salientar que a busca local não é utilizada em todas as subpopulações pois utilizar a mesma indiscriminadamente pode ocasionar uma perda de performance na busca global, pois problemas que tem espaço de busca muito extenso podem acabar tornando o processo de busca local oneroso para o algoritmo. Tendo este fato em mente, foi preferível adotar busca local em apenas uma subpopulação que neste algoritmo é chamada de *elite*, especializando-a na intensificação da busca e sendo assim a população que possui a maior probabilidade de encontrar a solução ótima do problema.

Enquanto a busca local é aplicada em apenas uma subpopulação, a mutação é aplicada em todas as outras exceto na elite. O método de mutação escolhido foi a Mutação Não-Uniforme detalhada no Capítulo 2 que também contribui para a adaptabilidade do algoritmo no processo evolutivo.

### 3.1.5 Evolução Sem Justiça

A evolução sem justiça não tem necessidade de se utilizar dos indicadores de domínio calculados na fase de consolidação pois a forma em que é estruturada a topologia das subpopulações em relação à migração não precisa fazer uso destes parâmetros para definir as rotas de migração, visto que estas rotas são definidas previamente de acordo com a literatura.

Nesta metodologia são utilizadas três topologias paralelas que se diferem nas rotas de migração. Mas no que diz respeito aos operadores evolutivos usados, estes são os mesmos utilizados na abordagem de evolução com justiça pois o fim deste trabalho é a análise e comparação dos resultados encontrados para cada topologia diferente utilizada.

É importante salientar que diferente da metodologia de evolução com justiça, esta abordagem não utiliza o mecanismo de geração contínua aleatória pois na evolução com justiça os indivíduos superiores são movidos da população para o *buffer* de saída para aguardar o momento da migração, não fazendo mais parte da subpopulação e consequentemente não participam da evolução. Na evolução sem justiça, os indivíduos superiores são apenas copiados para o *buffer* de saída, desta maneira os mesmos continuam na subpopulação e participam do processo evolutivo.

No que diz respeito à seleção de indivíduos para a etapa de cruzamento, a mesma também acontece junto com a seleção para migração, diferindo apenas no fato de que os indivíduos escolhidos como pais serão selecionados para o cruzamento independente de serem tidos como superiores ou não. Na evolução sem justiça o critério de superioridade

é dado pela Equação 3.4 que calcula o nível do *fitness* do indivíduo selecionado para cruzamento:

$$f_{nível} = \delta * (f_{ij}^- - f_{ij}^+) + f_{ij}^- + K \quad (3.4)$$

Onde  $\delta$  é um valor constante fixado em 0,2,  $f_{ij}^+$  é o *fitness* do melhor indivíduo da subpopulação,  $f_{ij}^-$  é o *fitness* do pior indivíduo e  $K$  é um valor constante muito pequeno fixado em  $10^{-6}$ . Esta equação mostra que o indivíduo para ser reconhecido como melhor, deve representar uma solução no mínimo 20% melhor que a melhor solução encontrada.

A definição das subpopulações para as quais os indivíduos escolhidos para a migrar serão enviados é feita antes da execução do algoritmo ao escolher a topologia de migração, funcionando da mesma forma que o modelo de competição justa exceto pelo fato de que as rotas de migração são definidas pela topologia e não pela qualidade do indivíduo a ser migrado.

### 3.1.6 Busca Local

A utilização de um mecanismo que intensifique a busca por soluções de qualidade em um algoritmo que trabalha com o melhoramento de uma população de soluções como o AG é essencial. A busca local no algoritmo proposto, é aplicada em uma única subpopulação que é denominada de subpopulação elite, fazendo com que o processo evolutivo não seja sobrecarregado computacionalmente e a busca esteja focada nas melhores soluções encontradas neste processo.

O método de busca local escolhido para este trabalho foi o algoritmo de *Hooke e Jeeves* que trabalha o melhoramento de soluções através de cálculos matemáticos básicos ocasionando uma boa convergência aliada à performance por fazer pouco uso de recursos computacionais.

Na abordagem de competição justa, a busca local é aplicada unicamente na população elite visto que de acordo com a topologia escolhida (HFC) a população elite recebe os melhores indivíduos advindos de todas as subpopulações. Esta característica é extremamente importante pois a subpopulação elite é uma subpopulação que representa uma região promissora no espaço de busca do problema, pois todas as melhores soluções encontradas no processo evolutivo fazem parte desta subpopulação então a probabilidade encontrar soluções muito boas ou até a solução ótima global ao aplicar um método de busca local para intensificar a busca, é muito alta.

Na abordagem de competição sem justiça a pesar de a subpopulação elite ter esta terminologia, na prática, não é trivial saber se ela representa uma região promissora pois os cálculos realizados nas etapas de pré-evolução e consolidação não são utilizados nesta abordagem. A utilização da busca local nesta subpopulação específica, para esta



abordagem, foi mantida por critérios de equidade nos mecanismos utilizados nas duas abordagens. Mas ainda assim, não exclui a probabilidade de a população elite realmente fazer jus à sua nomenclatura.

### 3.1.7 Migração

Em Algoritmos Evolutivos Paralelos, a migração é uma etapa muito importante pois é o momento onde as subpopulações trocam informações, enviando e recebendo indivíduos umas das outras de acordo com a topologia definida e política de migração.

No algoritmo proposto, além da população em que ocorre o processo de evolução existem duas outras populações auxiliares chamadas de *buffer* de saída e *buffer* de entrada. Estes *buffers* são de extrema importância para este algoritmo visto que são elas que definem o momento em que o processo de migração deverá ocorrer.

À medida que o processo evolutivo acontece, as subpopulações escolhem indivíduos que são tidos como os melhores de cada geração. Estes melhores indivíduos, são movidos ou copiados para o *buffer* de saída da subpopulação em que estão e, também, para o *buffer* de entrada da subpopulação a qual foi escolhida para que estes migrem. Em cada subpopulação o *buffer* de saída comporta todos os indivíduos que estão sendo enviados e o *buffer* de entrada comporta que estão sendo recebidos. O processo de migração ocorre no momento em que, ou o *buffer* de saída fica cheio, ou o *buffer* de entrada se esvazia.

Em todos os modelos de topologia que são abordados nos tópicos seguintes, o padrão de utilizar a mutação em todas as subpopulações com exceção da elite é adotado. A população elite no algoritmo proposto utiliza apenas busca local, como mecanismo de intensificação de busca. A migração ocorre entre subpopulações vizinhas e esta vizinhança entre as mesmas é definida pela topologia adotada, nas Subseções que seguem as topologias paralelas escolhidas para este trabalho são melhor detalhadas.

#### 3.1.7.1 Hierarchical Fair Competition

O modelo *Hierarchical Fair Competition* (HFC), é inspirado em processos de estratificação de classes que costumeiramente ocorrem em sociedades, e tem objetivo de preservar indivíduos com menor potencial de competirem com indivíduos com maior potencial. Esta estratificação ocorre criando uma espécie de hierarquia onde cada indivíduo se encaixa em uma classe hierárquica de indivíduos similares em relação ao seu *fitness* (HU et al., 2002 apud OLIVEIRA, 2004). Um exemplo de estratificação e hierarquia de classes pode ser encontrado na Figura

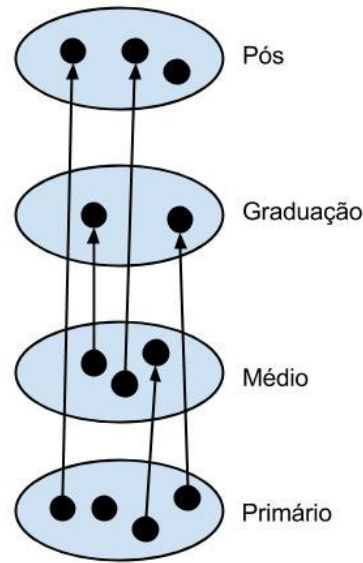


Figura 11 – Exemplo de estratificação em hierarquia de classes (OLIVEIRA, 2004)

Neste modelo as rotas de migração da topologia são definidas de modo que os indivíduos só podem migrar para subpopulações que sejam superiores em relação à subpopulação em que eles se encontram no momento da migração. A Figura 12 demonstra esta característica.

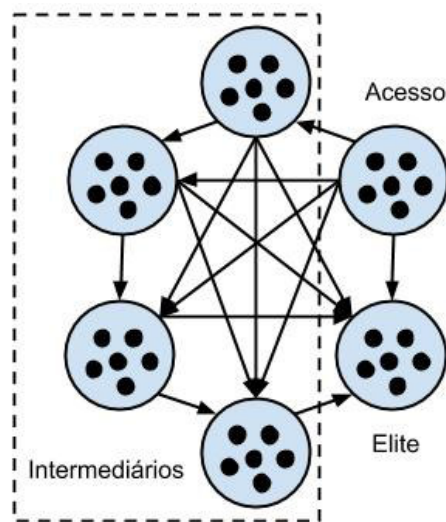


Figura 12 – Topologia do modelo HFC (OLIVEIRA, 2004)

De acordo com a Figura 12, pode-se perceber que existem três categorias de subpopulações:

1. Subpopulação de Acesso:

Nesta ocorre apenas o envio de indivíduos superiores e nunca o recebimento. Nela é utilizado o mecanismo de geração contínua de indivíduos, pois como ela apenas envia, há o risco de morte da subpopulação por falta de indivíduos.

## 2. Subpopulações Intermediárias:

Nestas ocorre o processo evolutivo normalmente, com todos os operadores evolutivos exceto busca local. A migração nestas subpopulações ocorre tanto no envio quanto no recebimento de indivíduos. Um detalhe importante é que nunca ocorre migração de indivíduos para subpopulações de níveis de *fitness* inferior a subpopulação em que o indivíduo se encontra no momento da migração.

## 3. Subpopulação Elite:

Aqui ocorre o recebimento de todos indivíduos superiores das subpopulações e nunca o envio, pois esta caracteriza o nível mais alto em que um indivíduo pode chegar de acordo com a proposta do modelo HFC. Na subpopulação elite não ocorre mutação mas em seu lugar e com mesma probabilidade, ocorre a busca local.

### 3.1.7.2 Ring

O modelo de topologia *Ring* é comumente usado na literatura em aplicações que visam manter populações evoluindo em paralelo. As rotas de migração deste modelo é definido neste trabalho da mesma forma em que é encontrado na literatura: os indivíduos devem migrar sempre para subpopulações vizinhas em apenas uma direção numa cadeia fechada de acordo com a Figura 13.

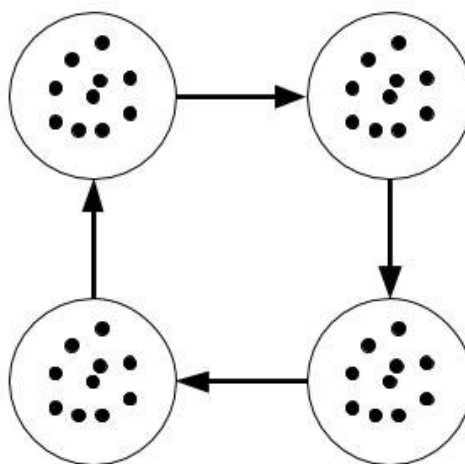


Figura 13 – Topologia de *Ring*

### 3.1.7.3 Ilhas

A topologia de Ilhas tem como principal característica uma ampla propagação de indivíduos entre as subpopulações. Nesta topologia qualquer subpopulação pode receber ou enviar indivíduos de e para outras subpopulações. A Figura 14 demonstra um esquema gráfico para a topologia de Ilhas.

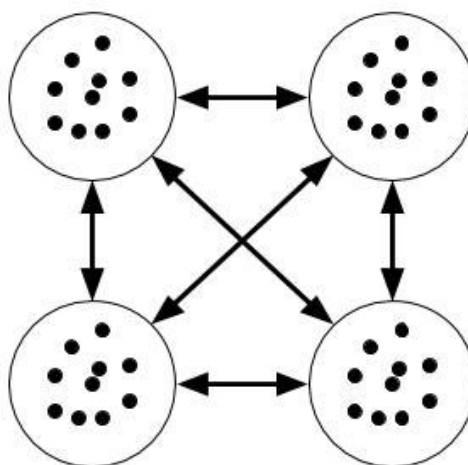


Figura 14 – Topologia de Ilhas

## 4 Experimentos Computacionais

Este capítulo dedica-se a apresentar os experimentos realizados e seus respectivos resultados.

### 4.1 Visão Geral

Os experimentos realizados neste estudo foram executados no super computador disponível no Centro Nacional de Processamento de Alto Desempenho. Foram utilizados os recursos de um nó de processamento, realizando experimentos com até oito unidades de processamento contidas no mesmo.

O experimento foi executado de acordo com o seguinte padrão de execução: Cada um dos quatro modelos deve ser executado para quatro configurações de subpopulações, cada subpopulação deve ser executada para as quatro funções de teste propostas, cada função deve ser executada vinte vezes, resultando em 910 jobs de execução submetidos ao super computador.

### 4.2 Resultados

Para analisar os resultados obtidos da execução foi utilizado o teste estatístico não-paramétrico de Kruskal-Wallis de amostras independentes. Este teste visa encontrar diferenças estatisticamente significativas entre pelo menos um par de grupos, entretanto, o mesmo não informa qual par possui diferença. Para descobrir quais pares possuem diferenças estatisticamente significativas, a comparação par a par vem a complementar o teste de Kruskal-Wallis como um teste *post hoc*.

O teste usado na comparação par a par foi o teste de Dunn-Bonferroni, onde o teste de Dunn aponta quais pares tiveram diferença estatisticamente significativa e, como há diversos testes sendo realizados entre os grupos, o valor de  $p$  precisa ser ajustado e neste ponto é utilizado o ajuste de Bonferroni (SPSS, 2016).

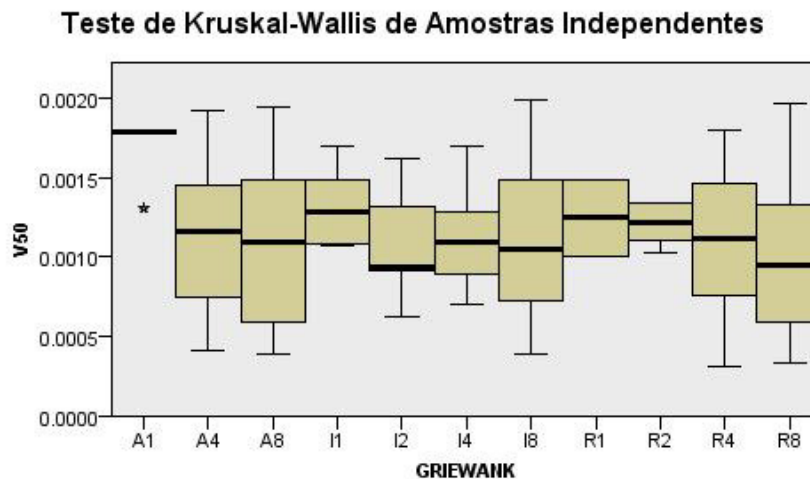
Para verificar se um par de grupos teve diferença estatisticamente significativa, utilizou-se o valor de  $p < 0,05$ , mostrando que com 95% de confiança, houve diferença entre os grupos analisados. Este limiar para o valor de  $p$  foi utilizado para definir quais combinações entre função e dimensão seriam abordadas neste tópico. Nas tabelas de resultado da comparação par a par este limiar assumiu também o papel de filtro, onde podem fazer parte da tabela somente os pares que possuem diferença estatisticamente significativa.

De acordo com a Tabela 1, a solução ótima de cada uma das funções deste trabalho é zero, com uma taxa de erro de 0,001. Dessa maneira todas as soluções encontradas que estão abaixo desse valor são consideradas ótimas.

## 4.2.1 Melhor Configuração de Subpopulação

### 4.2.1.1 Griewank com 50 variáveis

Como mostram a Figura 15 e a Tabela 2, I2 apresentou a melhor média enquanto R4 obteve a melhor solução. Todavia, em termos de significância estatística, de um modo geral, todos os métodos apresentaram desempenho semelhante entre si, com exceção do APHAC sequencial que teve desempenho inferior a todos os demais nesta função.



<b>N total</b>	220
<b>Estatística do teste</b>	47.464
<b>Graus de liberdade</b>	10
<b>Sig. assintótica (teste bilateral)</b>	.000

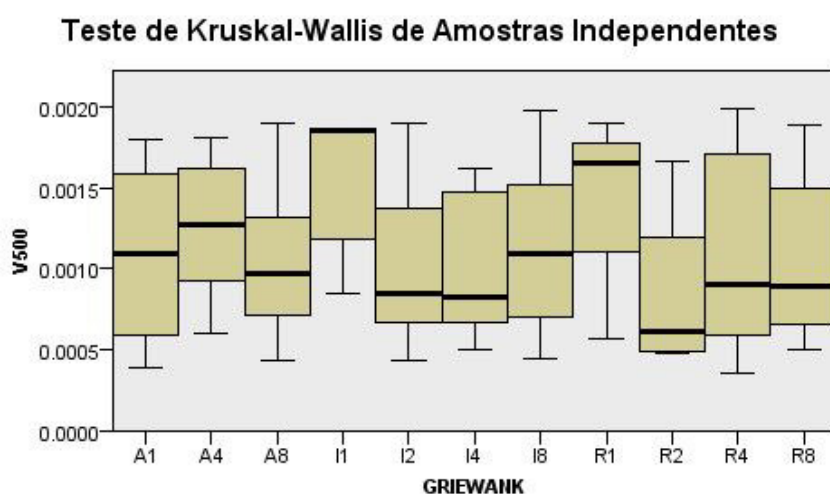
Figura 15 – Resultados para Griewank com 50 variáveis.

Tabela 2 – Resultado da comparação par a par em Griewank com 50 variáveis

Amostra1	Amostra2	$Média_{Amostra1}$	$Melhor_{Amostra1}$	$Média_{Amostra2}$	$Melhor_{Amostra2}$
A4	A1	0,001013654	0,000393095	0,001719776	0,00130937
A8	A1	0,001244979	0,000401792	0,001719776	0,00130937
R1	A1	0,001248375	0,00100759	0,001719776	0,00130937
R2	A1	0,001214012	0,00102366	0,001719776	0,00130937
R4	A1	0,00109716	0,000308571	0,001719776	0,00130937
R8	A1	0,001004535	0,000333526	0,001719776	0,00130937
I2	A1	0,001002328	0,000703178	0,001719776	0,00130937
I4	A1	0,001146815	0,000393095	0,001719776	0,00130937
I8	A1	0,001110427	0,000401792	0,001719776	0,00130937

#### 4.2.1.2 Griewank com 500 variáveis

Para Griewank com 500 variáveis, de acordo com a Tabela 3, o teste de comparação par a par mostrou que o melhor competidor foi o modelo R2. Este mostrou ter resultados estatisticamente significantes em relação aos modelos R1 e I1. O modelo I1 obteve piores resultados em comparação aos demais competidores. Ainda que R2 possa ser considerado o vencedor, o modelo que encontrou a melhor solução dentre todos os outros foi A1.



<b>N total</b>	220
<b>Estatística do teste</b>	37.898
<b>Graus de liberdade</b>	10
<b>Sig. assintótica (teste bilateral)</b>	.000

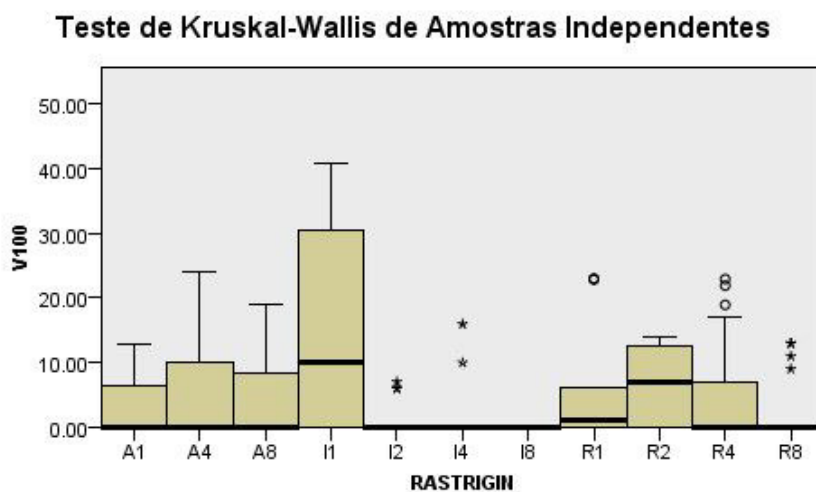
Figura 16 – Resultados para Griewank com 500 variáveis.

Tabela 3 – Resultado da comparação par a par em Griewank com 500 variáveis.

Amostra1	Amostra2	$Média_{Amostra1}$	$Melhor_{Amostra1}$	$Média_{Amostra2}$	$Melhor_{Amostra2}$
R2	R1	0,000843169	0,000480302	0,001444769	0,000563276
R2	I1	0,000843169	0,000480302	0,001626105	0,000850731
I4	I1	0,00098452	0,000498577	0,001626105	0,000850731
I2	I1	0,001035482	0,000430312	0,001626105	0,000850731
A1	I1	0,001091568	0,000385944	0,001626105	0,000850731
A8	I1	0,001040261	0,000429138	0,001626105	0,000850731
R8	I1	0,001086015	0,000496614	0,001626105	0,000850731

#### 4.2.1.3 Rastrigin com 100 variáveis

De acordo com o gráfico da Figura 17, os competidores I2, I4 e I8 obtiveram desempenho muito parecido, e nota-se pela Tabela 4, que não existe diferença estatisticamente significativa entre eles. A partir destes dados, pode-se dizer que o vencedor foi o competidor I8 que obteve a melhor média e, também, a melhor solução. Como mostra a Tabela 4, o competidor que obteve os piores resultados foi A1, onde até a melhor solução encontrada para o modelo foi pior que a média das soluções encontradas pelo vencedor.



<b>N total</b>	220
<b>Estatística do teste</b>	33.866
<b>Graus de liberdade</b>	10
<b>Sig. assintótica (teste bilateral)</b>	.000

Figura 17 – Resultados para Rastrigin com 100 variáveis.

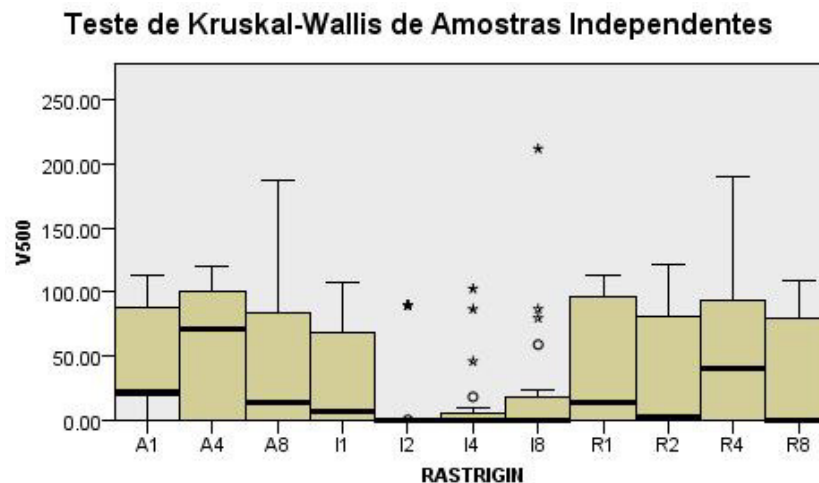


Tabela 4 – Resultado da comparação par a par em Rastrigin com 100 variáveis.

Amostra1	Amostra2	$Média_{Amostra1}$	$Melhor_{Amostra1}$	$Média_{Amostra2}$	$Melhor_{Amostra2}$
I8	A1	0,000634	6,38E-06	3,2352708	0,000858
I8	R1	0,000634	6,38E-06	4,8756298	9,7E-05
I8	R2	0,000634	6,38E-06	6,9648618	3,12E-05
I2	R1	0,945631	0,000388	4,8756298	9,7E-05
I2	R2	0,945631	0,000388	6,9648618	3,12E-05

#### 4.2.1.4 Rastrigin com 500 variáveis

Para Rastrigin com 500 variáveis o competidor I2 foi, em termos de significância estatística, o melhor dentre todos os outros competidores. De acordo com a Tabela 5, ocorreu diferença estatisticamente significativa apenas entre I2 e A1, onde este ultimo foi apontado como o pior competidor.



<b>N total</b>	220
<b>Estatística do teste</b>	25.316
<b>Graus de liberdade</b>	10
<b>Sig. assintótica (teste bilateral)</b>	.005

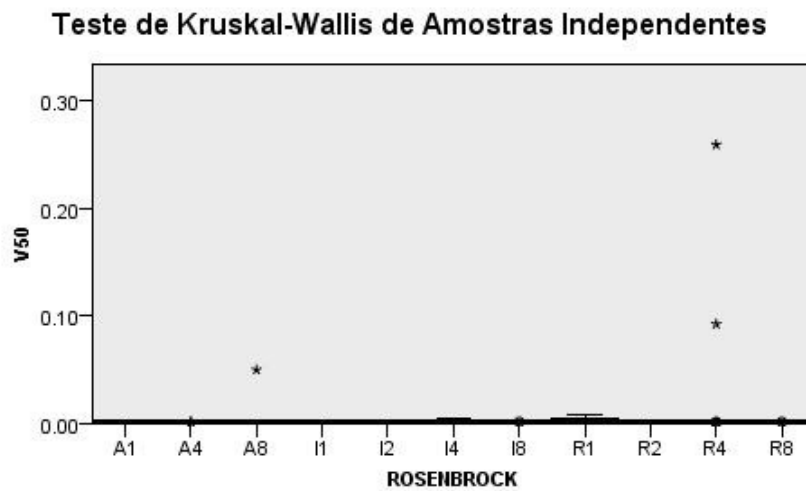
Figura 18 – Resultados para Rastrigin com 500 variáveis.

Tabela 5 – Resultado da comparação par a par em Rastrigin com 500 variáveis

Amostra1	Amostra2	$Média_{Amostra1}$	$Melhor_{Amostra1}$	$Média_{Amostra2}$	$Melhor_{Amostra2}$
I2	A1	17,91008	1E-05	46,11641	0,000437

#### 4.2.1.5 Rosenbrock com 50 variáveis

Como pode-se observar no gráfico da Figura 19, todos os competidores obtiveram um desempenho muito parecido, no que tange à significância estatística. Este fato fica explicitado com os dados apresentados na Tabela 6, que demonstra que só houve diferença estatisticamente significativa entre I1 e I4, sendo que o competidor I1 foi o vencedor pois obteve a melhor média e a melhor solução.



<b>N total</b>	220
<b>Estatística do teste</b>	19.547
<b>Graus de liberdade</b>	10
<b>Sig. assintótica (teste bilateral)</b>	.034

Figura 19 – Resultados para Rosenbrock com 50 variáveis.

Tabela 6 – Resultado da comparação par a par em Rosenbrock com 50 variáveis.

Amostra1	Amostra2	$Média_{Amostra1}$	$Melhor_{Amostra1}$	$Média_{Amostra2}$	$Melhor_{Amostra2}$
I1	I4	0,001506	0,001045	0,002532	0,001388

#### 4.2.1.6 Rosenbrock com 500 variáveis

Os competidores para Rosenbrock com 500 variáveis se saíram ligeiramente piores que os da análise com 50 variáveis mas, apesar disso, todos os competidores tiveram praticamente o mesmo desempenho. Este fato fica claro na Tabela 7, que mostra que houve diferenças estatisticamente significativas entre A1, A4 e R1, sendo que este ultimo foi o pior de todos. De acordo com a Tabela 7, A1 teve melhor média que A4 e R1, mas ainda assim A4 encontrou a melhor solução.

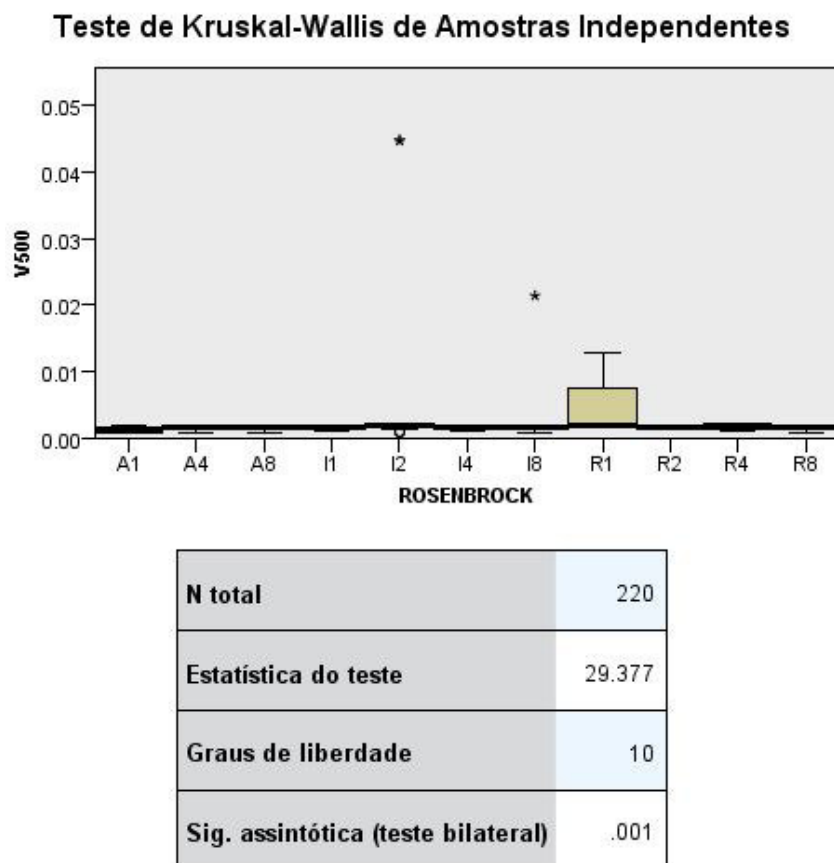


Figura 20 – Resultados para Rosenbrock com 500 variáveis.

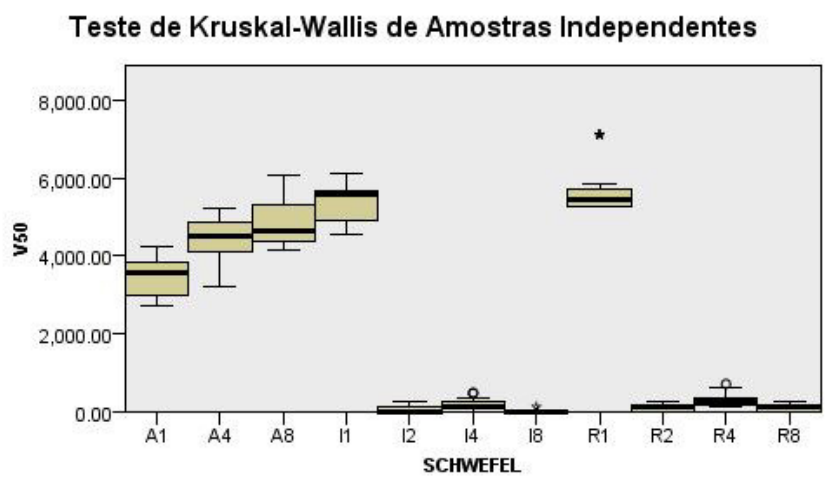
Tabela 7 – Resultado da comparação par a par em Rosenbrock com 500 variáveis.

Amostra1	Amostra2	$Média_{Amostra1}$	$Melhor_{Amostra1}$	$Média_{Amostra2}$	$Melhor_{Amostra2}$
A1	R1	0,001278	0,000886	0,004597	0,001709
A4	R1	0,001519	0,000858	0,004597	0,001709

#### 4.2.1.7 Schwefel com 50 variáveis

Para a função Schwefel com 50 variáveis, todos os competidores tiveram performance ruim em relação a média, entretanto o competidor R1 obteve o pior resultado. De acordo com o gráfico da Figura 21 e a Tabela 8, todos os competidores que utilizaram apenas uma subpopulação obtiveram péssimas soluções. Em relação à significância estatística, percebe-se que os competidores I2, I4, I8, R2, R4 e R8 foram os melhores e, dentre eles, o competidor I2 obteve a melhor média e a melhor solução.

Os competidores da topologia APHAC tiveram os piores resultados, e esta baixa performance se deve ao fato de que apesar de o modelo HFC dividir o espaço de busca uniformemente entre as subpopulações, pode ocorrer de certas faixas de *fitness* serem mapeadas para subespaços de busca extensos, causando ineficiência na busca (OLIVEIRA, 2004).



<b>N total</b>	220
<b>Estatística do teste</b>	193.468
<b>Graus de liberdade</b>	10
<b>Sig. assintótica (teste bilateral)</b>	.000

Figura 21 – Resultados para Schwefel com 50 variáveis.

Tabela 8 – Resultado da comparação par a par em Schwefel com 50 variáveis.

Amostra1	Amostra2	$Média_{Amostra1}$	$Melhor_{Amostra1}$	$Média_{Amostra2}$	$Melhor_{Amostra2}$
I8	R4	5,922672	9,9055E-06	307,9399	118,438
I8	A1	5,922672	9,9055E-06	3438,715	2704,36
I8	A4	5,922672	9,9055E-06	4430,672	3217,59
I8	A8	5,922672	9,9055E-06	4852,139	4145,45
I8	I1	5,922672	9,9055E-06	5453,274	4579,69
I8	R1	5,922672	9,9055E-06	5774,045	5290,34
I2	A1	65,14159	3,639E-06	3438,715	2704,36
I2	A4	65,14159	3,639E-06	4430,672	3217,59
I2	A8	65,14159	3,639E-06	4852,139	4145,45
I2	I1	65,14159	3,639E-06	5453,274	4579,69
I2	R1	65,14159	3,639E-06	5774,045	5290,34
R8	A1	82,9069	3,8589E-05	3438,715	2704,36
R8	A4	82,9069	3,8589E-05	4430,672	3217,59
R8	A8	82,9069	3,8589E-05	4852,139	4145,45
R8	I1	82,9069	3,8589E-05	5453,274	4579,69
R8	R1	82,9069	3,8589E-05	5774,045	5290,34
R2	A1	76,98516	1,506E-05	3438,715	2704,36
R2	A4	76,98516	1,506E-05	4430,672	3217,59
R2	A8	76,98516	1,506E-05	4852,139	4145,45
R2	I1	76,98516	1,506E-05	5453,274	4579,69
R2	R1	76,98516	1,506E-05	5774,045	5290,34
I4	A4	165,8136	0,00035362	4430,672	3217,59
I4	A8	165,8136	0,00035362	4852,139	4145,45
I4	I1	165,8136	0,00035362	5453,274	4579,69
I4	R1	165,8136	0,00035362	5774,045	5290,34
R4	A8	307,9399	118,438	4852,139	4145,45
R4	I1	307,9399	118,438	5453,274	4579,69
R4	R1	307,9399	118,438	5774,045	5290,34

#### 4.2.1.8 Schwefel com 500 variáveis

Para Schwefel com 500 variáveis, percebe-se que todas as topologias obtiveram péssimos resultados. Apesar deste baixo desempenho, nota-se que o competidor A1 foi o perdedor em termos de significância estatística em detrimento aos outros competidores arrolados na Tabela 9. Pode-se perceber que vencedor da competição foi R2 pois obteve a melhor média, entretanto o competidor que obteve a melhor solução foi I1.

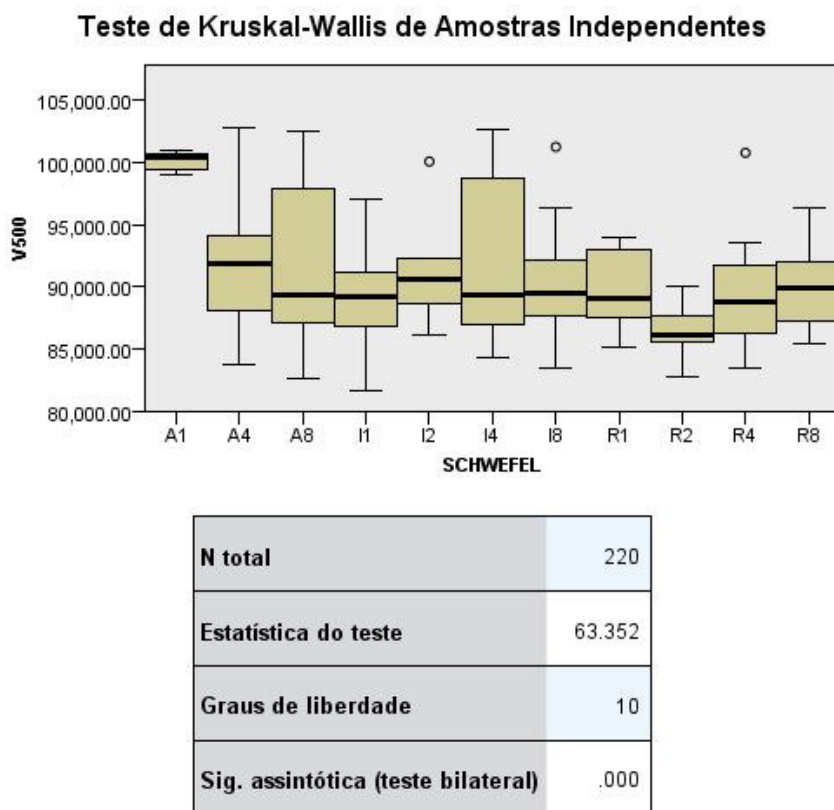


Figura 22 – Resultados para Schwefel com 500 variáveis.

Tabela 9 – Resultado da comparação par a par em Schwefel com 500 variáveis.

Amostra1	Amostra2	$Média_{Amostra1}$	$Melhor_{Amostra1}$	$Média_{Amostra2}$	$Melhor_{Amostra2}$
R2	I2	86497,48	82742,6	90610,78	86055,1
R2	I4	86497,48	82742,6	92234,99	84362,6
R2	A4	86497,48	82742,6	92461,77	83688,8
R2	A1	86497,48	82742,6	100223,6	99047
I1	A1	88848,12	81629,2	100223,6	99047
R4	A1	89287,27	83528,1	100223,6	99047
R1	A1	89548,26	85146,3	100223,6	99047
I8	A1	90162,21	83452,2	100223,6	99047
R8	A1	90075,51	85426,4	100223,6	99047
A8	A1	91709,63	82577	100223,6	99047
I2	A1	90610,78	86055,1	100223,6	99047
I4	A1	92234,99	84362,6	100223,6	99047
A4	A1	92461,77	83688,8	100223,6	99047

### 4.3 Speedup

Nesta seção, os tempos de execução das topologias para 50 e 500 variáveis são analisados visando avaliar o *speedup* dos algoritmos paralelos apresentados. O *speedup* é o fator de redução de tempo de execução à medida em que um programa é executado

paralelamente em  $p$  processadores. Pode-se dizer que trata-se da relação entre o tempo sequencial e o tempo paralelo de um dado algoritmo para resolver um dado problema em uma máquina específica (ROCHA, 2008).

Nesta análise, são levados em conta os tempos de execução para as topologias APHAC, *Ring* e Ilhas. Cada topologia variando-se o número de subpopulações de 1 a 8, aplicadas às funções *Griewank*, *Rastrigin*, *Rosenbrock* e *Schwefel*.

É importante ressaltar que para a topologia APHAC não foram executados experimentos com duas subpopulações, pois o modelo de competição justa requer no mínimo 3 subpopulações (acesso, intermediária e elite) para sua execução. Por este motivo, o tempo de execução com 1 subpopulação foi utilizado para substituir o valor correspondente a 2 subpopulações.

O tempo de execução total usado para cálculo do *speedup* corresponde ao tempo de execução de um dado nó para alcançar a solução ótima, considerando a margem de erro já mencionada de 0,001.

#### 4.3.1 Griewank com 50 variáveis

A função *Griewank* sendo executada com 50 variáveis mostra, através do gráfico na Figura 23, que as topologias R e I têm um bom desempenho quando executam com duas subpopulações, quando este número cresce o desempenho cai. Na topologia A verifica-se que com 4 subpopulações o desempenho é bom, mas com 8 há uma ligeira queda de desempenho.

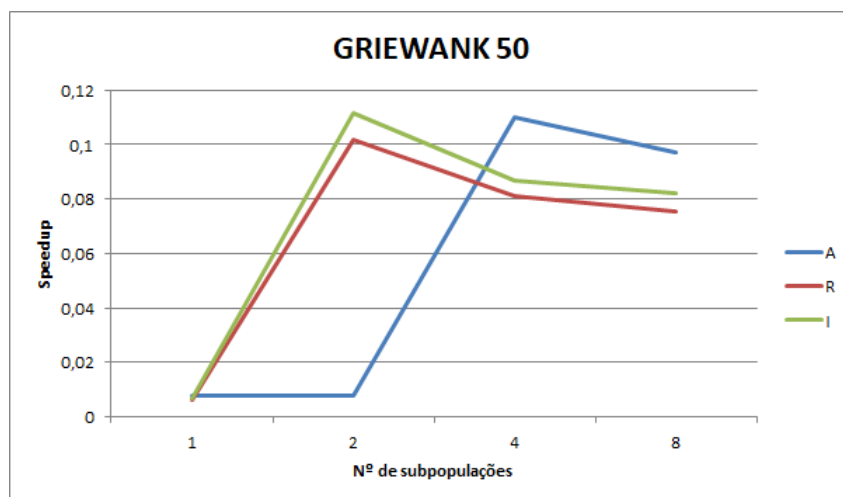


Figura 23 – Speedup para Griewank com 50 variáveis.

### 4.3.2 Griewank com 500 variáveis

No gráfico da Figura 24, que a topologia R tem bom desempenho a medida que o número de subpopulações cresce até 4, após isso há uma queda de desempenho. Na topologia I o aumento do número de subpopulações não significa aumento de desempenho, pois a medida que este numero cresce o desempenho varia entre bom e ruim. Para a topologia A o desempenho cresce com 4 subpopulações e descrece com 8.

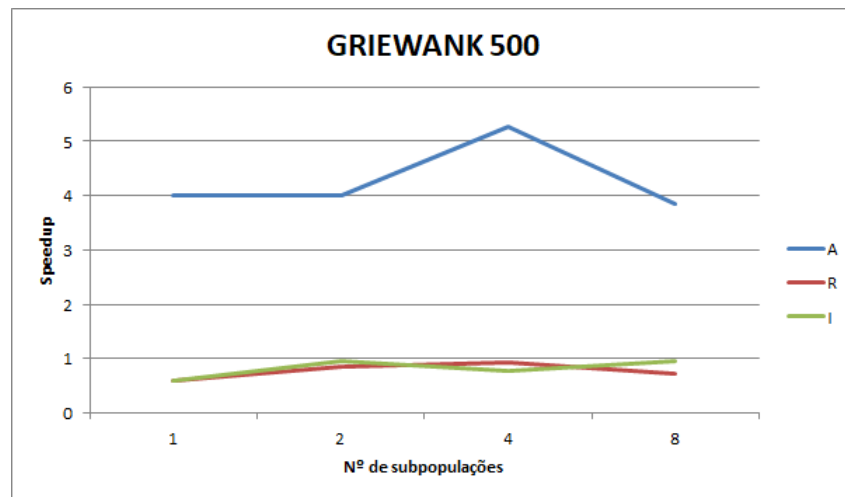


Figura 24 – Speedup para Griewank com 500 variáveis.

### 4.3.3 Rastrigin com 100 variáveis

Neste cenário, a Figura 25 mostra que houve uma queda brusca no desempenho quando as topologias executaram com 2 subpopulações, e quando este número foi incrementado, houve um pequeno aumento no desempenho das topologias R e I, mas este não foi significativo.

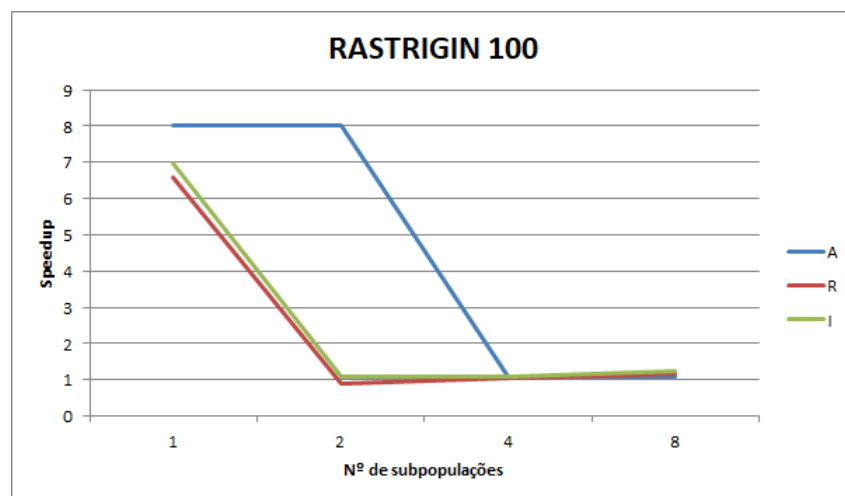


Figura 25 – Speedup para Rastrigin com 100 variáveis.



#### 4.3.4 Rastrigin com 500 variáveis

Assim como na análise da Figura 25, o cenário de *Rastrigin* com 500 variáveis possui características semelhantes. Todas as topologias apresentam baixíssimo desempenho ao executar com um maior número de subpopulações, e a Figura 26 demonstra isso.

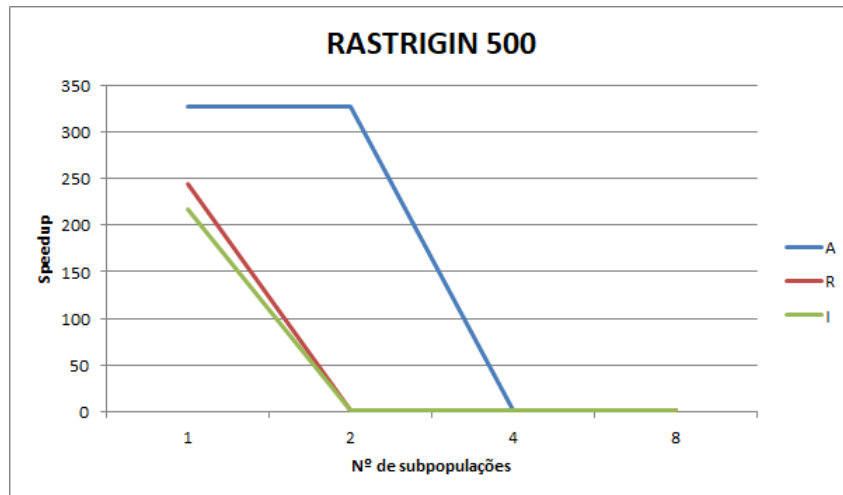


Figura 26 – Speedup para Rastrigin com 500 variáveis.

#### 4.3.5 Rosenbrock com 50 variáveis

A Figura 27, mostra que ao aumentar o número de subpopulações, o desempenho das topologias R e I aumentou quando utiliza 2 subpopulações e decaiu quando o número de subpopulações foi maior que 2. Na topologia A, o desempenho aumentou com 4 subpopulações e teve uma queda quando executou com 8.

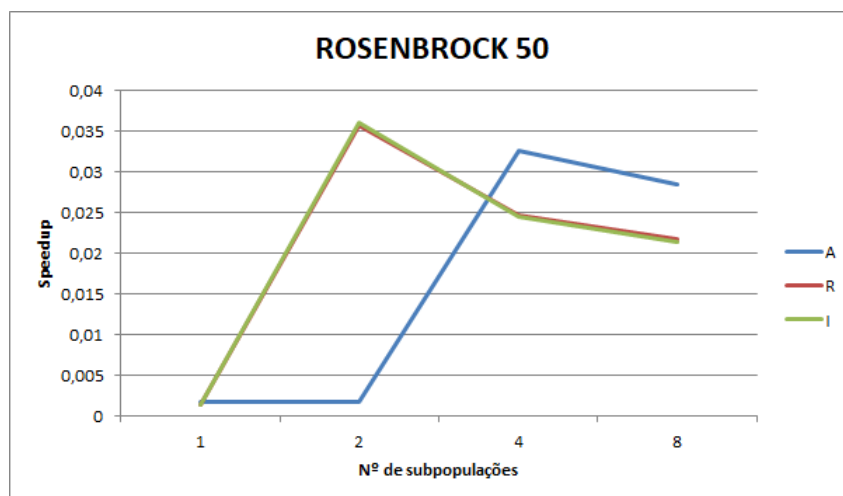


Figura 27 – Speedup para Rosenbrock com 50 variáveis.

### 4.3.6 Rosenbrock com 500 variáveis

Para Rosenbrock com 500 variáveis, o desenvolvimento da topologia I é de aumento de desempenho com 2 subpopulações, e queda de desempenho com número maior que este. Já na topologia A, houve aumento de desempenho com 4 subpopulações e queda com 8. A topologia R teve um desempenho razoável pois teve uma crescente de desempenho a medida que o número de subpopulações aumentou até 4, após isso houve queda no desempenho.

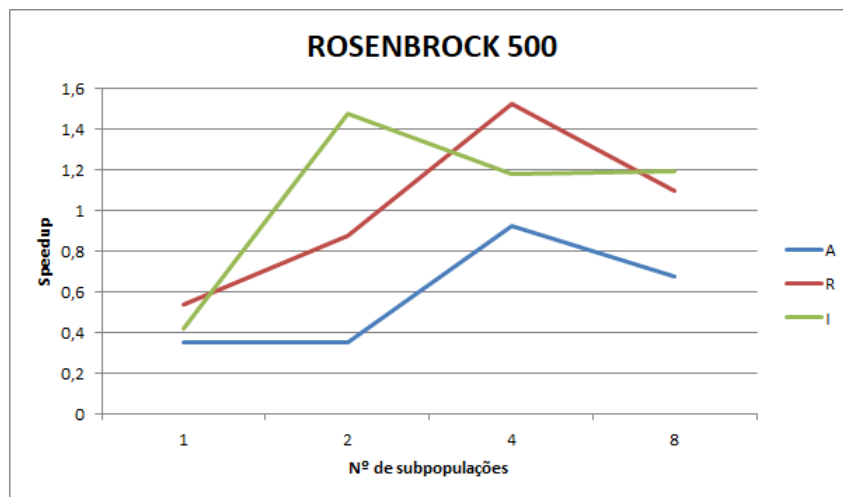


Figura 28 – Speedup para Rosenbrock com 500 variáveis.

### 4.3.7 Schwefel com 50 variáveis

De acordo com a Figura 29, todas as topologias tiveram péssimo desempenho com o aumento do número de subpopulações.

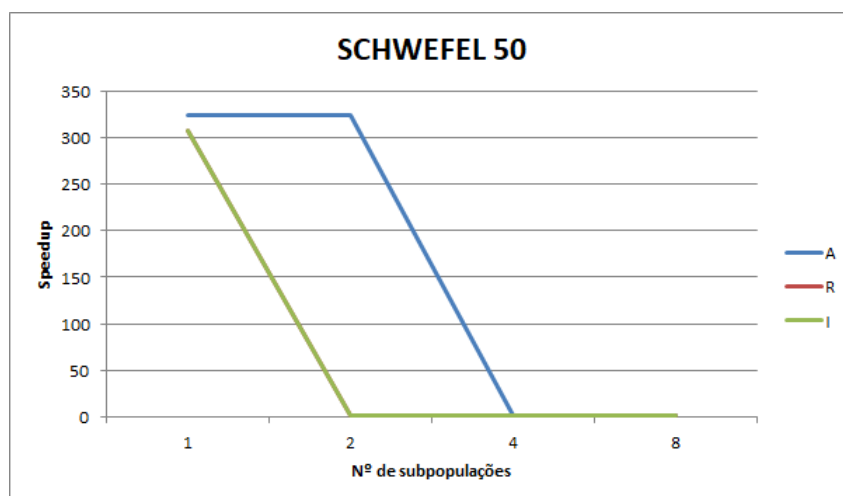


Figura 29 – Speedup para Schwefel com 50 variáveis.

### 4.3.8 Schwefel com 500 variáveis

Assim como na Figura 29 o cenário do gráfico da Figura 30 é bem semelhante, há uma brusca queda de desempenho quando o número de subpopulações aumenta.

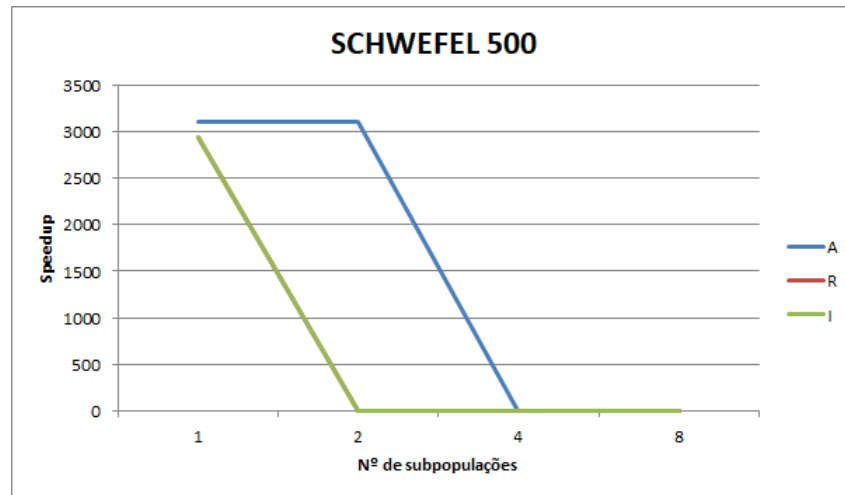


Figura 30 – Speedup para Schwefel com 500 variáveis.

## 4.4 Discussão

Os resultados obtidos sugerem que algoritmos evolutivos, de um modo geral, têm boa capacidade de minimizar funções irrestritas de muitas variáveis contínuas, alcançando as soluções ótimas com taxas de acerto significativas. Os resultados também indicam que as versões paralelas processadas em dois nós têm desempenho melhor que as versões executando em oito nós. Essa queda no desempenho observada em todos os modelos de paralelismo usados pode ocorrer por dois motivos. Primeiramente, devido a algum tipo de configuração da conta no servidor CENAPAD que pode levar a um escalonamento de processos que executa melhor versões sequenciais do que versões paralelas. Ou seja, a versão sequencial é multi-processada e sem troca de mensagens, enquanto a versão paralela pode perder eficiência em decorrência da troca de mensagens e/ou compartilhamento de processadores com processos de *background* da biblioteca MPI. Além disso, o gerenciamento de *buffers* de transferência de indivíduos das versões paralelas podem não ter sido implementados da melhor forma.

## 5 Conclusão

Algoritmos Paralelos Híbridos são abordagens contemporâneas usadas satisfatoriamente na resolução de problemas de otimização de grande porte, pois podem tirar proveito de hardwares sofisticados de alto desempenho, permitindo redução de tempo de execução e aumento significativo de eficiência.

O algoritmo proposto neste trabalho foi desenvolvido na linguagem C em conjunto com a biblioteca *Message Passing Interface*. Este possui especialização nas topologias APHAC, Ring e Ilhas, tendo potencial para englobar outras topologias mais. É importante salientar que o algoritmo possui uma implementação da topologia Stepping Stones, porém esta não pôde ser efetivamente testada para poder ser analisada neste trabalho.

Neste trabalho foram utilizadas funções numéricas irrestritas como forma de avaliação do algoritmo proposto e suas topologias. Mediante a execução de cada topologia do algoritmo no super computador disponível no CENAPAD-UFC, foi possível observar que a topologia APHAC foi a pior em 7 de 8 análises. A topologia Ilhas foi vencedora com 5 vitórias em 8 análises, enquanto APHAC obteve 1 vitória em 8, sendo esta a pior topologia. Quanto ao número de subpopulações, os testes executados com 2 subpopulações obtiveram 5 vitórias em 8 análises, enquanto as outras configurações de subpopulação obtiveram 1 vitória cada. Quando executados com apenas 1 subpopulação, os testes falharam em 7 de 8 análises. Em relação ao desempenho em tempo de execução foi possível perceber que para os cenários analisados, o aumento do número de subpopulações não resultou efetivamente em aumento de desempenho, principalmente com número de subpopulações superior a 4.

### 5.1 Trabalhos Futuros

Como trabalhos futuros, pretende-se dar continuidade a implementação do *framework* para adicionar novas topologias e novas funções objetivos, relativas a outros problemas com apelo mais realista.

Efetuar mais execuções do algoritmo para cada topologia, aumentando ainda mais o número de subpopulações e a dimensionalidade das funções utilizadas para validar o bom funcionamento do mesmo.

Estudar os motivos do baixo desempenho ao incrementar o número subpopulações e saná-los. Implementar novos mecanismos de gerenciamento de *buffers* de transferência de indivíduos.

Pode-se estudar métodos para lidar com o problema da ineficiência do algoritmo quando faixas de *fitness* de subespaços de busca muito grandes são atribuídas às subpopu-

lações, para que o desempenho do algoritmo possa melhorar gradativamente.

# Referências

- BAZARAA, M. S.; JARVIS, J. J.; SHERALI, H. D. *Linear Programming and Network Flows (2Nd Ed.)*. New York, NY, USA: John Wiley & Sons, Inc., 1990. ISBN 0-471-63681-9. Citado na página 29.
- BESSAOU, M.; SIARRY, P. A genetic algorithm with real-value coding to optimize multimodal continuous functions. *Springer-Verlag*, 2001. Citado na página 18.
- BLICKLE, T. *A Comparison of Selection Schemes Used in Genetic Algorithms*. [S.l.], 1995. Citado 2 vezes nas páginas 18 e 20.
- CORTES, O. A. C.; SILVA, J. C. d. A local search algorithm based on clonal selection and genetic mutation for global optimization. In: *2010 Eleventh Brazilian Symposium on Neural Networks*. [S.l.: s.n.], 2010. p. 241–246. ISSN 1522-4899. Citado na página 29.
- COSTA, R. A. L. *Um Implementação Paralela do Algoritmo de Evolução Diferencial Autoadaptativo*. 2010. Monografia (Bacharel em Ciência da Computação), Universidade Federal de Ouro Preto, Ouro Preto, Brasil. Citado na página 27.
- COUTO, D. C. F.; SILVA, C. A.; BARSANTE, L. S. Otimização de funções multimodais via técnica de inteligência computacional baseada em colônia de vaga-lumes. *CILAMCE Proceedings of the XXXVI Iberian Latin-American Congress on Computational Methods in Engineering*, 2015. Citado na página 14.
- CUNHA, V. H.; CAMPOS, E. de S.; GUIMARÃES, L. de C.; DANTAS, M. J. P. Algoritmo genético de codificação real aplicado à otimização de funções de benchmark. *Simpósio Brasileiro de Pesquisa Operacional*, 2016. Citado na página 18.
- DEAP. *Benckmarks*. 2012. <<http://deap.gel.ulaval.ca/doc/0.7/api/benchmarks.html>>. Citado 4 vezes nas páginas 9, 30, 31 e 32.
- GRAMA, A.; GUPTA, A.; KARYPIS, G.; KUMAR, V. *Introduction to Parallel Computing*. [S.l.]: Addison Wesley, 2003. Second Edition. Citado na página 27.
- GROSAN, C.; ABRAHAM, A. Hybrid evolutionary algorithms: Methodologies, architectures, and reviews. In: \_\_\_\_\_. *Hybrid Evolutionary Algorithms*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 1–17. ISBN 978-3-540-73297-6. Disponível em: <[https://doi.org/10.1007/978-3-540-73297-6\\_1](https://doi.org/10.1007/978-3-540-73297-6_1)>. Citado 3 vezes nas páginas 9, 25 e 26.
- HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. [S.l.]: University of Michigan Press, 1975. Citado na página 22.
- HOOKE, R.; JEEVES, T. A. “ direct search ” solution of numerical and statistical problems. *J. ACM*, ACM, New York, NY, USA, v. 8, n. 2, p. 212–229, abr. 1961. ISSN 0004-5411. Disponível em: <<http://doi.acm.org/10.1145/321062.321069>>. Citado na página 33.
- HU, J.; GOODMAN, E. D.; SEO, K.; PEI, M. *Adaptive hierarchical fair competition (AHFC) model for parallel evolutionary algorithms*. New York: Morgan Kaufmann Publishers, 2002. 772–779 p. Citado 2 vezes nas páginas 27 e 40.

- JEBARI, K. Selection methods for genetic algorithms. v. 3, p. 333–344, 12 2013. Citado 2 vezes nas páginas 18 e 19.
- JUNIOR, E. de C. R.; COSTA, J. P.; CORTES, O. A. C. A parallel adaptative genetic algorithm for unconstrained multimodal numerical optimization. *XIII Simpósio Brasileiro de Automação Inteligente*, 2017. Citado na página 29.
- LIU, C. *Multi-Robot Task Allocation for Inspection Problems with Cooperative Tasks Using Hybrid Genetic Algorithms*. [S.l.: s.n.], 2014. ISBN 3862195511. Citado na página 25.
- MEHRA M.L. JAYALAL, A. J. A. S. R. K. K. S. S. M. M. Study on different crossover mechanisms of genetic algorithm for test interval optimization for nuclear power plants. v. 6, p. 20–28, 01 2014. Citado na página 23.
- NOWOSTAWSKI, M.; POLI, R. *Parallel genetic algorithm taxonomy*. Adelaide: IEEE Press, 1999. 88-92 p. Citado na página 29.
- OLIVEIRA, A. C. M. de. *Algoritmos Evolutivos Híbridos com Detecção de Regiões Promissoras em Espaços de Busca Contínuos e Discretos*. Tese (Doutorado) — INPE, São José dos Campos, 2004. Citado 14 vezes nas páginas 9, 14, 16, 17, 20, 21, 27, 28, 29, 30, 36, 40, 41 e 50.
- PESSINI, E. C. *Algoritmos Genéticos Paralelos – Uma Implementação Distribuída Baseada em JavaSpaces*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Florianópolis, 2003. Citado na página 28.
- ROCHA, M. L. *APLICAÇÕES DE ALGORITMOS PARALELOS E HÍBRIDOS PARA O PROBLEMA DE ÁRVORE DE STEINER EUCLIDIANA NO R*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2008. Citado na página 54.
- SHUKLA, A.; PANDEY, H.; MEHROTRA, D. Comparative review of selection techniques in genetic algorithm. 02 2015. Citado na página 20.
- SPSS. *Kruskal-Wallis Test in SPSS*. 2016. <[https://www.sheffield.ac.uk/polopoly\\_fs/1.714567!/file/step-marshall-KruskalSPSS.pdf](https://www.sheffield.ac.uk/polopoly_fs/1.714567!/file/step-marshall-KruskalSPSS.pdf)>. Citado na página 44.
- ZHAO, X.; GAO, X.-S. Evolutionary programming based on non-uniform mutation. 12 2004. Citado na página 24.