RODOLFO AYALA LOPES COSTA

Orientador: Frederico Gadelha Guimarães

UM IMPLEMENTAÇÃO PARALELA DO ALGORITMO DE EVOLUÇÃO DIFERENCIAL AUTOADAPTATIVO

Ouro Preto

Novembro de 2010

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

UM IMPLEMENTAÇÃO PARALELA DO ALGORITMO DE EVOLUÇÃO DIFERENCIAL AUTOADAPTATIVO

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

RODOLFO AYALA LOPES COSTA

Ouro Preto
Novembro de 2010

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

Um Implementação Paralela do Algoritmo de Evolução Diferencial Autoadaptativo

RODOLFO AYALA LOPES COSTA

Monografia defendida e aprovada pela banca examinadora constituída por:

Dr. Frederico Gadelha Guimarães – Orientador
Universidade Federal de Minas Gerais

Dr. Joubert de Castro Lima
Instituto Tecnológico de Aeronáutica

Dr. Marcone Jamilson Freitas Souza
Universidade Federal do Rio de Janeiro

Ms. Ricardo Sérgio Prado
Universidade Federal de Minas Gerais

Ouro Preto, Novembro de 2010

Resumo

A Otimização e Inteligência Computacional tem sido as áreas que recebem um maior número de contribuições a partir de técnicas de computação evolutiva nos últimos tempos. Esta abordagem de computação evolutiva, deve ser compreendida como um conjunto de técnicas e procedimentos genéricos e adaptáveis para serem empregados a solução de problemas complexos. Este trabalho propõem a implementação e análise dos resultados obtidos do algoritmo de Evolução Diferencial Autoadaptativo em paralelo, utilizando a esquema conhecido como modelo de ilhas, ou seja, múltiplas subpopulações com o intuito de obter melhor desempenho do método de otimização.

Abstract

The optimization and computational intelligence have got a lot of contributions from the techniques evolutionary computation over the years. This approach of the evolutionary computation must be understand how a collection of the generics procedures and adaptable techniques to be use to solve complexes problems. This work proposes the implementation and analyses of the results from the algorithmic Self-Adaptive Differential Evolution in parallel, making use of schema known as island model, i.e., multiple subpopulations in order to achieve better performance of the optimization method.

Dedico este trabalho a minha família que sempre me apoiou e acompanhou nos bons e maus momentos. Amo todos vocês!

Agradecimentos

Agradeço, primeiramente, a Deus, que sempre me guiou e abençou todos os dias.

Agradeço aos meus pais, por todo amor e confiança depositada.

Agradeço ao Professor Fred, pela oportunidade de ser seu orientando e, apesar do tão pouco tempo de convivênvia, pela confiança em minha capacidade de realização deste trabalho.

Agradeço aos meus familiares e aos velhos e bons amigos que torceram e rezaram por mim nesta longa caminhada.

Agradeço aos grandes amigos de República, por bons momentos vividos que vou levar comigo pelo resto da vida.

Agradeço aos meus amigos da turma de Ciência da Computação, pela a amizade e companheirismo durante estes três anos de convivênvia, principalmente ao Rodrigo pelo grande apoio na realização deste trabalho.

Enfim, a todos que fizeram parte desta caminhada meus sinceros agradecimentos!

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

Ao longo da história, a indústria tecnológica vem pesquisando continuamente como aumentar o poder computacional dos atuais computadores, na expectativa de atender a demanda de mercado que está sempre a procura de uma maior capacidade de processamento, não importando quão alto este poder tenha alcançado. A principal relevância deste esforço é ostentar aplicações que demandam alto processamento de informações e alto desempenho.

Seguindo esta linha de pesquisa surgiu a tecnologia de Processamento Paralelo com intuito de dividir uma tarefa grande em pequenas tarefas de menor custo computacional, buscando solucionar o problema proposto de forma mais rápida ou um problema de complexidade maior com o mesmo tempo computacional. Em afirmação a estas ideias, a procura por maior desempenho, menor custo e produtividade sustentada, foi a principal justificativa nos últimos anos para uma crescente aceitação e adoção do processamento paralelo, tanto para computação de alto desempenho quanto para aplicações científicas, devido à facilidade de se encontrar processadores paralelos e o uso generalizado de computação distribuída [Tasoulis *et al.*, 2004].

Existem inúmeras aplicações com diversos fins que fazem o uso das técnicas de processamento paralelo e computação distribuída. Um bom exemplo deste tipo de aplicação e sem fins lucrativos é o World Community Grid que reúne membros do mundo inteiro criando um grid de computadores, que são os recursos computacionais de vários computadores não pertencendo à uma única organização, sendo gerenciado por sistema que faz alocação das tarefas. O propósito é agregar poder de processamento dos computadores de cada um dos membros para realizar pesquisas humanitárias que demandariam uma enorme infraestrutura computacional [World Community Grid, 2010].

A comunidade possui atualmente cerca de 526000 membros espalhados por todos os continentes do mundo, como mostra a Figura 1.1. Com esta estrutura gigantesca que ultrapassa não apenas as barreiras intermunicipais ou interestaduais, mas sim as barreiras intercontinentais conectados entre si pela grande rede, a internet, é possível alcançar um poder computacional até então inimaginável, podendo ser utilizado em benefício das grandes organizações ou em prol de toda a humanidade.

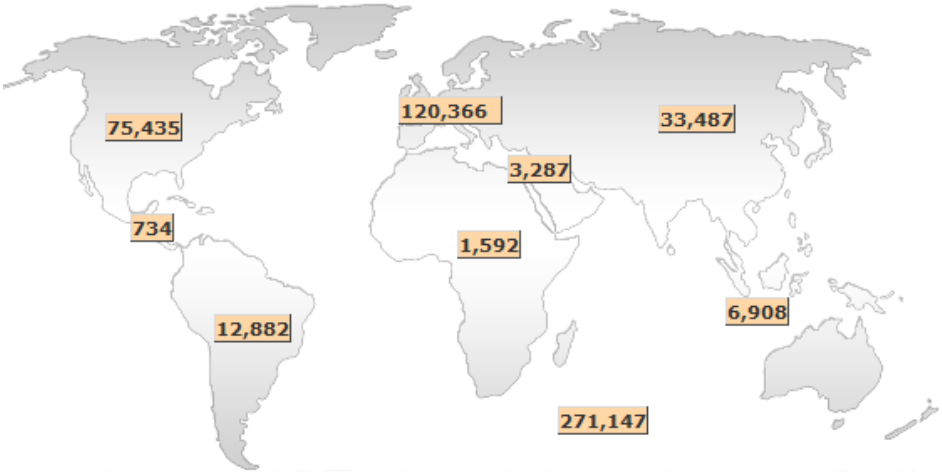

Figura 1.1: Mapa apresentando o número de membros da comunidade por continente e membros que não informaram sua localização. Fonte: World Community Grid

Distante da Computação Paralela, mas com objetivos ligeiramente semelhantes, que possuem como principal meta o melhor gerenciamento de recursos, na qual nós últimos anos vem sendo assistido pelo uso de computadores, está as técnicas de Pesquisa Operacional.

A Pesquisa Operacional teve seu primeiro ápice durante a 2^a Guerra Mundial entre os anos de 1939 a 1945, onde os comandos das tropas aplicavam um método científico para gerenciar os gastos de recursos raros às tropas em combate. Anos depois, estes métodos foram incorporados às indústrias da época para assistir o planejamento e controle da produção industrial de forma mais eficiente, reduzindo a escassez de matéria prima, desperdícios e aumentando o lucro.

Existem várias técnicas para resolução de problemas operacionais como Otimização Linear, Otimização Não-linear, Otimização Dinâmica, Otimização Inteira entre muitas outras técnicas. Sendo assim, vários problemas práticos e científicos podem ser solucionados pelas técnicas de otimização. Estas abordagens podem facilmente encontrar uma boa solução para problemas como o do Roteamento de Veículos, de Escala de Tripulante [Silva *et al.*, 2007], de montar a Programação de Jogos de Competições Esportivas [Souza *et al.*, 2006], problema de Controle de Pátio de Minérios [Moraes *et al.*, 2006] e muitos outros desafios.

Outra área de estudo que tem mobilizado a investigação por novas propostas ou melhorias, são as técnicas bioinspiradas da Otimização e Inteligência Computacional, pois problemas que envolvem a otimização global sobre espaços contínuos estão onipresentes em toda a comunidade científica [Storn and Price, 1997], além de que, quando bem empregadas podem auxiliar na tomada de decisões dos processos empresariais. Assim, inerente a este segmento encontramos as classes de algoritmos chamados de evolutivos, que são técnicas inspiradas no princípio Darwiniano da evolução das espécies e na genética [Goldberg, 1989].

Inserido no contexto de Computação Evolutiva nos deparamos com diversas famílias de

métodos, como os Algoritmos Genéticos (GA) [Goldberg, 1989], Sistemas Imunes Artificiais (AIS) [Castro, 2002], Otimização por enxame de partículas (PSO) [Poli *et al.*, 2007] e Evolução Diferencial (DE) [Storn and Price, 1995].

O algoritmo de Evolução Diferencial é um otimizador simples para funções em espaços contínuos não lineares e não diferenciáveis, sendo a função unimodal ou multimodal, proposto em 1995 por (Storn and Price, 1997). Este otimizador é robusto, fácil de usar e se adapta bem a paralelização, que o torna muito valioso, devido à necessidade de desempenho [Storn and Price, 1997]. Ele requer a configuração de alguns parâmetros, mas os resultados experimentais indicam que o método possui boas propriedades de convergência e supera outros métodos de algoritmos evolutivos [Storn and Price, 1997], [Storn, 1999].

Todavia, no algoritmo de Evolução Diferencial clássico existem alguns parâmetros de controle que devem ser definidos antes que o processo de otimização se inicie. No entanto, o processo de encontrar os melhores valores para estes parâmetros não é bem conhecido e nem muito determinístico [Liu and Lampinen, 2005]. Estes parâmetros possuem importância crucial no desempenho do algoritmo, além do mais, é dependente da função a qual se deseja otimizar [Brest and Maucec, 2006].

Da necessidade de se configurar os parâmetros de controle e escolher a estratégia de mutação surgiram as abordagens adaptativas e autoadaptativas para o método DE. O algoritmo de Evolução Diferencial Autoadaptativo permite que os parâmetros de controle se adaptem a qualquer classe de problemas no qual o algoritmo esteja lidando, reconfigurando-se automaticamente sem qualquer intervenção do usuário.

Portanto, tendo em vista esses tópicos inovadores e vantajosos, mas em segmentos completamente diferentes que a computação incitou nestes últimos tempos, este projeto visa unir estes campos para desenvolver e analisar o desempenho de um algoritmo de Evolução Diferencial Autoadaptativo Paralelo em proveito próprio de ambos.

Sendo o principal objetivo aplicar uma estratégia de paralelização para o método DE conhecida na literatura ao algoritmo de Evolução Diferencial Autoadaptativo, como o esquema denominado de modelo de ilhas, que são subpopulações de indivíduos aleatórios onde há troca de informações entre as ilhas por meio do processo de migração, causando uma cooperação entre elas de forma que a evolução de uma auxilie na evolução da outra.

1.1 Motivação e relevância

Os problemas de otimização podem ser descritos como problemas computacionais em que o objetivo é encontrar uma boa solução entre todas as soluções em um espaço finito ou infinito de possíveis valores, ou seja, encontrar um bom valor de mínimo ou máximo de uma dada função objetivo. Como a maioria dos problemas em diferentes áreas de estudo podem ser formulados como problemas de otimização, a necessidade de um algoritmo de otimização eficiente se faz

presente em toda comunidade científica [Milani and Santucci, 2010].

Dentre as principais abordagens de algoritmos para otimização, existe a classe de algoritmos evolutivos que apresenta várias famílias de métodos, a qual se encontra o algoritmo de Evolução Diferencial. Sua disseminação vem sendo explicada pelo fato de ser um poderoso otimizador global, no contexto de otimização mono objetivo [Mezura-Montes *et al.*, 2006; Chakraborty, 2008] e, mais recentemente, multiobjetivo [Mandavan, 2002; Li *et al.*, 2009], além de ser bastante robusto.

Entretanto, o algoritmo de Evolução Diferencial possui problemas, podendo apresentar dificuldade em se chegar a convergência quando imposto a problemas de maior complexidade, ou seja, em funções-objetivo mais complexas, tornando o método mais lento [Tomassini, 1999]. Se não bastasse, este método tem a necessidade que algumas variáveis de controle sejam configuradas antes de sua execução, sendo que elas estão intrinsecamente ligadas ao desempenho do algoritmo, e a escolha dos melhores valores não é um processo muito bem conhecido [Liu and Lampinen, 2005].

Assim, a proposta é implementar o algoritmo de Evolução Diferencial Autoadaptativo de parâmetros em paralelo, com o intuito de diminuir o número de gerações que a abordagem sequencial utiliza para convergir, fator muito importante quando se trata de algoritmos evolutivos. Contudo, o desenvolvimento da paralelização deste algoritmo é de grande valia, pois conduziria a um melhor desempenho do método de Evolução Diferencial Autoadaptativo que além de mais simples é mais eficiente que o algoritmo clássico.

1.2 Objetivos

Este trabalho tem como principal meta implementar o algoritmo de Evolução Diferencial Autoadaptativo em paralelo, usando para tal a estratégia de paralelização conhecida como modelo de ilhas para algoritmos genéticos[Cantú-Paz, 1998]. Este modelo gera subpopulações de indivíduos aleatórios, onde há troca de informações entre as subpopulações por meio do processo de migração, causando uma cooperação entre elas.

Desta forma, temos como objetivos específicos estudar o desempenho do algoritmo de Evolução Diferencial Autoadaptativo sequencial, avaliando principalmente a quantidade de gerações que o método utiliza para convergência. Estudar as abordagens propostas na literatura para paralelização do método, focado principalmente no modelo de ilhas, além de conhecer um pouco mais sobre as políticas de migração para este modelo.

Finalizar o trabalho implementando e analisando o desempenho do algoritmo de Evolução Diferencial Autoadaptativo em paralelo, avaliando principalmente o número de gerações para se chegar à convergência, comparado com sua abordagem em sequencial.

1.3 Fundamentação teórica

Esta seção apresenta alguns trabalhos que contribuíram para a fundamentação teórica deste.

1.3.1 A evolução diferencial

Segundo Storn e Price (1997) o algoritmo de evolução diferencial é uma nova abordagem de otimização para funções espaciais contínuas, que requer algumas variáveis de controle, mas é robusto, fácil de usar e se adapta bem a computação paralela.

1.3.2 A evolução diferencial autoadaptativa

A adaptação, em especial autoadaptação, tem sido empregada para que o ajuste dos parâmetros de controle sejam autoadaptados ao problema a qual esteja sendo tratado, sem que haja intervenção do usuário [Brest *et al.*, 2006]. Desta maneira, o usuário não necessita conhecimento algum do método que esta utilizando.

1.3.3 O modelo de ilhas

Segundo Tasoulis *et al.* (2004) o modelo de ilhas se caracteriza pelo uso de subpopulações, onde cada uma está associada a um processo e evoluem de forma independente em direção a uma solução. Para promover a troca de informações existe o processo de migração, onde indivíduos migram de uma subpopulação para outra.

1.4 Estrutura e organização do trabalho

Este trabalho está estruturado da seguinte forma, o primeiro capítulo introduz os conceitos fundamentais ao seu entendimento, assim como trabalhos correlatos. O segundo capítulo contextualiza os problemas de otimização e apresenta o algoritmo de Evolução Diferencial clássico, demonstrando suas principais características. São descritos os 3 (três) operadores empregados no método que são: mutação diferencial, recombinação e seleção. O conceito e trabalhos relacionados sobre autoadaptação de parâmetros são introduzidos, assim como, uma proposta para autoadaptação do algoritmo.

O terceiro capítulo refere-se à paralelização do algoritmo de Evolução Diferencial, sendo feita uma apresentação das técnicas de computação paralela, das estratégias de paralelização do método, principalmente o modelo de ilhas. Neste capítulo, são introduzidas também as características do modelo de ilhas, da mesma forma que, nossa abordagem proposta para paralelização utilizando esse modelo.

O quarto capítulo apresenta os resultados alcançados e avaliação das abordagens sequencial e paralela, comparando essencialmente, o número de gerações que cada abordagem demora a

convergir. Uma avaliação do *SpeedUp*, para a abordagem paralela é exposta, com o intuito de avaliar o desempenho em termos de tempo computacional, dessa abordagem.

Para concluir, o último capítulo faz uma análise da paralelização, utilizando o modelo de ilhas para o algoritmo de Evolução Diferencial, com a finalidade de avaliar o quão é compensador o uso dessa estratégia, neste contexto específico.

Capítulo 2

Evolução Diferencial

A Teoria da Evolução pela seleção natural, também conhecida como Darwinismo, é a grande inspiração para a computação evolutiva. Ela consiste na aplicação das ideias oriundas dessa teoria como modelo para resolução de problemas através de computadores, pois se acredita que a seleção natural explica o desaparecimento de indivíduos não adaptados ao meio, causada pelas alterações genéticas não ótimas. Desta forma, podemos enxergar a evolução biológica como uma potencial ferramenta para o processo de otimização aplicada à minimização ou maximização dos recursos que desejamos potencializar.

A maior virtude da computação evolutiva é não ter a carência de indicar os passos necessários que conduzem a um resultado, pois o simples fato de descrever matematicamente o que se deseja na solução já torna possível a resolução do problema. Devido a este fato, tem-se acompanhado nestes últimos tempos um aumento do domínio de aplicações que fazem o uso dessas ferramentas, além de melhoras no desempenho delas em vários aspectos.

Existem dentro da Computação Evolutiva diversas famílias de métodos, tais como os algoritmos Genéticos, os Sistemas Imunes Artificiais, a Otimização por Exame de Partículas e o algoritmo de Evolução Diferencial, entre outros. Os algoritmos Genéticos (GA) são métodos probabilísticos que possuem mecanismos de buscas paralela e adaptativa baseado no darwinismo [Pacheco, 1999]. Os Sistemas Imunes Artificiais (AIS) são algoritmos exploratórios das características de aprendizagem e memória dos sistemas imunes [Castro, 2002]. A Otimização por Exame de Partículas (PSO) é um método onde uma série de partículas são colocadas no espaço de busca de um problema e cada uma avalia sua função objetivo na sua atual localização [Poli *et al.*, 2007]. E a Evolução Diferencial (DE), a qual será o foco deste capítulo, sendo abordada mais adiante, bem como uma variação deste método para autoadaptação de parâmetros.

2.1 Características da Evolução Diferencial

O algoritmo de Evolução Diferencial proposto por Storn e Price em 1995 [Storn and Price, 1995] é um poderoso otimizador, apresentado inicialmente para resolver problemas de otimização não lineares com variáveis contínuas, como já mencionado anteriormente. Atualmente é um importante método no contexto de otimização mono objetivo [Mezura-Montes *et al.*, 2006; Chakraborty, 2008], e nos últimos tempos tem sido adaptado para resolução de problemas multi-objetivo, sendo que sua primeira versão foi proposta por Chang *et al.* (1999) com muito sucesso.

Devido ao desempenho e versatilidade do método, que lhe confere grande aplicação e adoção diante de problemas práticos de diversas áreas como, por exemplo, a otimização de sistemas de reservatório [Reddy and Kumar, 2007], projeto de filtros digitais [Storn, 1999], além de ser usado para treinamento de redes neurais artificiais de forma muito eficiente [Masters and Land, 1997; Magoulas *et al.*, 2001; Abbass, 2002; Ilonen *et al.*, 2003]. Sendo consolidado como um método de otimização de propósito geral.

Como em todo algoritmo evolutivo, o DE faz uso de uma população de soluções candidatas para o problema, no intuito de explorar o espaço de busca. Uma população é compreendida em NP indivíduos representados como um vetor de variáveis $x_{i,G}$, $i = 1, 2, 3, ..., NP$ e G é a geração corrente a qual o individuo $x_{i,G}$ pertence. A quantidade de indivíduos de uma população não sofre alterações durante o processo de otimização, sendo que a população inicial é escolhida de forma aleatória seguindo uma distribuição uniforme. Como dito, a representação de cada indivíduo na população corrente é feita através de um vetor de variáveis de otimização que possui a dimensão D, $x_{i,G} = [x_{i,G,1}, x_{i,G,2}, ..., x_{i,G,D}]$.

A ideia fundamental do algoritmo é empregar um operador de mutação diferencial com a intenção de gerar perturbações nos indivíduos da população corrente, como podemos ver na Figura 2.1, que será comparada ao indivíduo corrente, fazendo, dessa forma, uma busca pelo espaço de soluções. Esse operador é inspirado em argumentos matemáticos e heurísticos. Isso demonstra que apesar deste método ser considerado um algoritmo evolutivo, o operador de mutação diferencial não possui nenhuma inspiração oriunda de um processo natural, mas é referenciado como tal, pelo fato do método conduzir a evolução da população de soluções candidatas.

À medida que o algoritmo progride, a aplicação do operador de mutação diferencial aos pares de indivíduos da população corrente gera vetores diferenciais que conduzem a uma modificação da distribuição espacial da população de acordo com a função objetivo do problema. Este fato é garantido pelos três operadores que o algoritmo de evolução diferencial apresenta: mutação, recombinação e seleção [Storn and Price, 1997]. Desta forma, os indivíduos experimentais gerados através da mutação e recombinação são selecionados pelo operador de seleção na qual serão escolhidos os indivíduos que irão estar presentes na próxima geração.

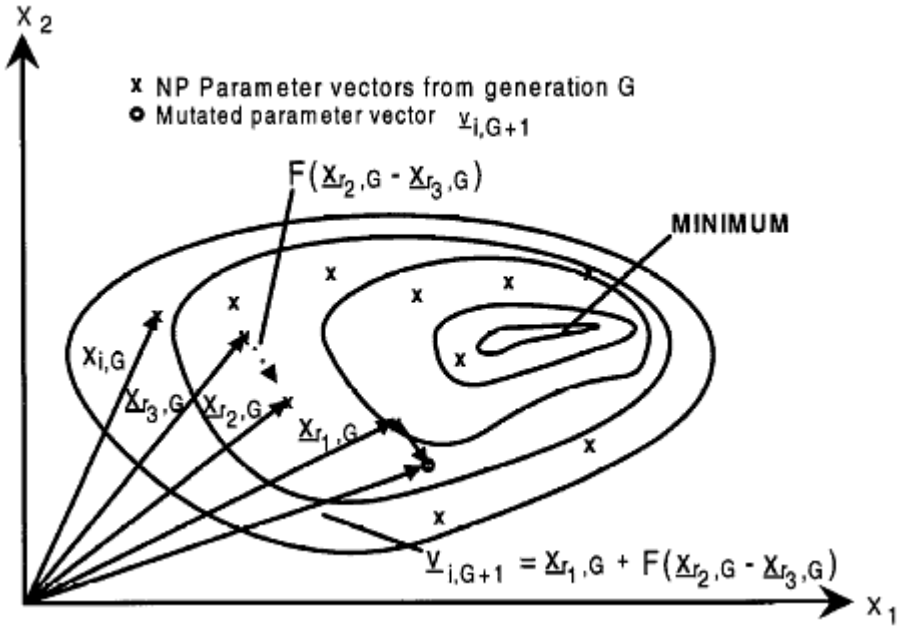

Figura 2.1: Exemplo de uma função de custo bidimensional mostrando suas linhas de contorno. Retirando de [Storn and Price, 1997].

2.1.1 Mutação

Para cada indivíduo da população corrente, um vetor mutante, também denominado de vetor diferencial, é gerado, empregando alguma das estratégias de mutação, sendo que as mais corriqueiras são:

- rand

$$v_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G}) \quad r1 \neq r2 \neq r3 \neq i \tag{2.1}$$

- best

$$v_{i,G+1} = x_{best,G} + F(x_{r2,G} - x_{r3,G}) \quad r2 \neq r3 \neq i \tag{2.2}$$

- current to best

$$v_{i,G+1} = x_{i,G} + F(x_{best,G} - x_{i,G}) + F(x_{r2,G} - x_{r3,G}) \quad r2 \neq r3 \neq i \tag{2.3}$$

- current to rand

$$v_{i,G+1} = x_{i,G} + F(x_{r1,G} - x_{i,G}) + F(x_{r2,G} - x_{r3,G}) \quad r1 \neq r2 \neq r3 \neq i \tag{2.4}$$

Onde $r1$, $r2$ e $r3$ são índices escolhidos aleatoriamente, mas diferentes entre si e diferentes de i, que representa o índice da solução candidata corrente. *Best* é o índice da melhor solução

dentro daquela população de soluções candidatas. De acordo com Storn and Price (1997), F é um fator real e constante no intervalo $[0, 2]$, que controla a amplificação da variação da diferença entre os vetores diferença.

Podemos observar que a estratégia *rand* utiliza apenas vetores selecionados aleatoriamente e diferentes entre si, sendo que esta estratégia é oriunda do algoritmo de evolução diferencial clássico. A *best* usa vetores aleatórios em seu operador de mutação, bem como o vetor que representa a melhor solução da população corrente. A *current-to-rand* utiliza vetores escolhidos aleatoriamente, além do vetor corrente. No entanto a estratégia *current-to-best*, além de utilizar os vetores escolhidos aleatoriamente e o vetor contendo a melhor solução da população corrente, faz também, o uso do vetor corrente no operador de mutação.

2.1.2 Recombinação

O operador de recombinação, também denominado operador de cruzamento, possui uma probabilidade CR de recombinação entre o indivíduo atual e o vetor diferencial, como podemos observar na Figura 2.2 que segue o seguinte padrão:

$$u_{i,G+1} = u_{1i,G+1}, u_{2i,G+1}, u_{3i,G+1}, ..., u_{Di,G+1} \tag{2.5}$$

onde

$$u_{ji,G+1} = \begin{cases} \mathrm{v}_{ji,G+1}, & se\ rand <= CR\ ou\ j = r \\ \mathrm{x}_{ji,G+1}, & se\ rand > CR\ e\ j \neq r \end{cases} \tag{2.6}$$

para $j = 1, 2, 3, ..., D$,$r \in (1, 2, 3, ..., D)$ e $rand \in (0, 1]$.

Dessa forma, seguindo este padrão é possível notar que pelo menos uma variável sofre recombinação.

2.1.3 Seleção

Na seleção para sobrevivência cada solução teste $u_{i,G}$ é comparada com a solução correspondente na população corrente $x_{i,G}$. Caso $u_{i,G}$ seja melhor que $x_{i,G}$, a solução teste substitui a solução corrente, caso contrário, a solução teste é eliminada.

$$x_{i,G+1} = \begin{cases} \mathrm{u}_{i,G}, & se\ f(u_{i,G+1}) < f(x_{i,G}) \\ x_{i,G}, & otherwise \end{cases} \tag{2.7}$$

Assim, este processo tem continuidade até que um critério de parada definido pelo usuário seja atendido.

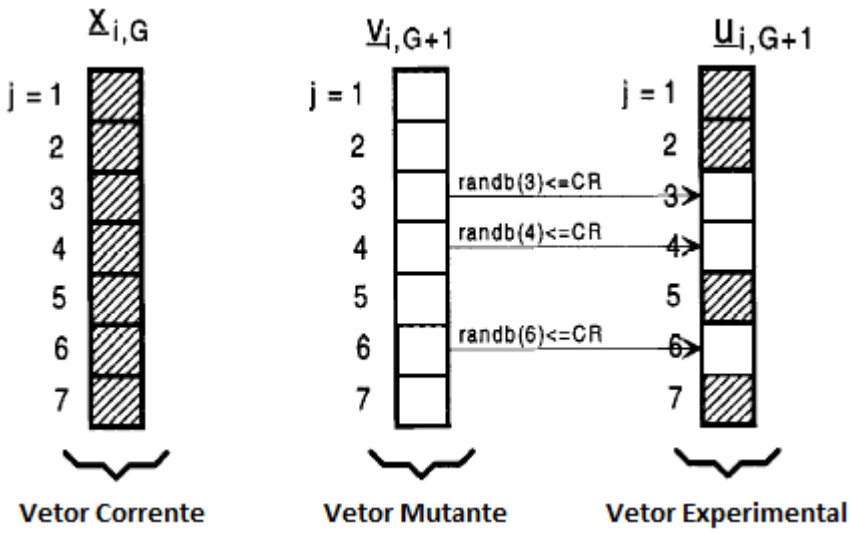

Figura 2.2: Ilustração do processo de recombinação para um vetor de dimensão igual a 7. Adaptado de [Storn and Price, 1997].

2.2 Evolução Diferencial Autoadaptativa

No algoritmo de evolução diferencial clássico o tamanho da população, o fator que controla a amplificação da variação da diferença entre os vetores diferença e a probabilidade de recombinação são parâmetros do método que devem ser definidos antes que o processo de otimização se inicie. No entanto, o processo de encontrar os melhores valores para estes parâmetros não é bem conhecido e nem muito determinístico [Liu and Lampinen, 2005]. Estes parâmetros possuem importância crucial no desempenho do algoritmo, além do mais, são dependentes da função na qual deseja otimizar [Janez and Mirjam, 2006].

A fixação destes parâmetros é geralmente conduzida por experimentos empíricos que determinaram os melhores valores para essas variáveis, processo este que demanda enorme tempo de dedicação e esforço do usuário. Desta forma, a ideia principal da autoadaptação de parâmetros é consentir que o método se autoadapte ao problema que esta sendo otimizado, através do conhecimento que irá adquirir durante o processo, sem que haja intervenção do usuário.

Alguns projetos relativos à autoadaptação na Evolução Diferencial já foram divulgados [Brest and Maucec, 2006; Qin and Suganthan, 2005; Teo, 2006]. No entanto, estes realizam autoadaptação de forma equívoca, pois presumimos que os parâmetros que estão sendo autoadaptatos devam ser avaliados de maneira implícita pelo operador de seleção do algoritmo, de forma a adquirir conhecimento sobre novos valores para os parâmetros do problema em questão.

Qin and Suganthan (2005) desenvolveram a primeira proposta que realmente seguia as ideias de autoadaptação do algoritmo de Evolução Diferencial Autoadaptativo (SADE - Self

Adaptive Differential Evolution), sendo que anos depois apresentaram uma versão melhorada do método [Qin *et al.*, 2009].

A característica fundamental do SADE é utilizar múltiplas estratégias de mutação, desta forma, além do parâmetro F (fator de amplificação) haverá um parâmetro CR (probabilidade de recombinação) e um valor p referente à probabilidade de aplicação para cada estratégia. Neste caso, a representação do indivíduo em estudo será da seguinte forma:

$$x_i = \langle x_{i,1}, x_{i,2}, x_{i,3}, ..., x_{i,D}, F_i, CR_i^1, CR_i^2, CR_i^3, CR_i^4, p_i^1, p_i^2, p_i^3, p_i^4 \rangle \tag{2.8}$$

notando que $\sum_{k=1}^{4} p_i^k = 1$.

2.2.1 Abordagem Proposta para Autoadaptação

A proposta é utilizar o algoritmo de Evolução Diferencial, prevendo o uso de múltiplas estratégias assim como o SADE, onde os parâmetros escolhidos para autoadaptação são os parâmetros de F (fator de amplificação) e CR (probabilidade de recombinação), considerando que o parâmetro NP (tamanho da população) não deve ser autoadaptado. Assim, as estratégias empregadas são: *best* e *rand*.

O parâmetro NP não interfere diretamente no valor de cada solução, assim não consideramos o NP uma boa escolha para autoadaptação. Portanto, para modificarmos o tamanho da população são necessários os parâmetros de todos os indivíduos, fazendo deste, um processo de adaptação, pois é feito através de uma retroalimentação ($feedback$) do processo.

Sendo assim, o indivíduo em análise x_i será representado da seguinte forma:

$$x_i = \langle x_{i,1}, x_{i,2}, x_{i,3}, ..., x_{i,D}, V_i^1, V_i^2, F_i^1, CR_i^1, F_i^2, CR_i^2 \rangle \tag{2.9}$$

onde $x_{i,d}$, são as variáveis de otimização e $V_i^k \in [0, 1]$ são os valores atribuídos a cada uma das estratégias utilizadas, sendo que cada estratégia possui um par de parâmetros F e CR.

Todos os parâmetros F, CR e V são selecionados aleatoriamente no intervalo $[0, 1]$ para a geração da população inicial. Em cada geração alteramos inicialmente os valores de V para cada indivíduo, empregando a própria mutação diferencial como se segue:

$$V_i^k = V_i^k + F'(V_{r1}^k - V_i^k) + F'(V_{r2}^k - V_{r3}^k) \tag{2.10}$$

onde o $F' \in [0, 0.5]$ e é escolhido aleatoriamente para cada indivíduo da população. Feito isso, escolhe-se a estratégia de mutação através da seleção por roleta, onde cada estratégia ocupa uma porção da roleta proporcional a sua probabilidade de aplicação.

Após a escolha da estratégia procedemos à alteração dos parâmetros da mesma, seguindo o mesmo esquema de mutação diferencial:

$$F_i^k = F_i^k + F'(F_{r1}^k - F_i^k) + F'(F_{r2}^k - F_{r3}^k) \tag{2.11}$$

$$CR_i^k = CR_i^k + F'(CR_{r1}^k - CR_i^k) + F'(CR_{r2}^k - CR_{r3}^k) \tag{2.12}$$

Em seguida, procedemos à mutação e o cruzamento referente à estratégia selecionada, utilizando os parâmetros codificados nos indivíduos. Mas em caso de extrapolação do valor no processo de mutação para os parâmetros F, CR e V, onde $F \in [0.1, 1]$, CR e $V \in [0, 1]$, seu valor é fixado no limite extrapolado.

Notemos que a realização de alteração dos valores de V (valores atribuídos a cada estratégia) e a escolha de qual estratégia será empregada antes de aplicarmos a mutação para os parâmetros F e CR, garantindo que sejam modificados somente os valores de F e CR que serão realmente avaliados. Assim, é possível garantir que a alteração de todos os parâmetros antes da aplicação das operações seja realmente avaliada implicitamente pelo operador de seleção. Desta maneira, temos um algoritmo de evolução diferencial que emprega de forma correta e em ordem coerente a autoadaptação de parâmetros.

Capítulo 3

Paralelismo e o Modelo de Ilhas

O mundo é essencialmente paralelo, ou seja, ao mesmo tempo em que neste momento, existem milhões de pessoas trabalhando, muitos outros eventos estão simultaneamente acontecendo, sem que haja qualquer influência direta de nossa vontade. Da mesma forma é possível aplicar o paralelismo à computação, com o desígnio de proporcionar economia de tempo e dinheiro para resolver problemas realmente grandes, onde a computação sequencial não ofereceria devido a suas limitações.

Assim sendo, a computação paralela tem sido acompanhada de uma crescente adoção, explicada pelo fato de ser uma ótima estratégia, devido à relação custo/benefício, para solucionar problemas de aplicações que demandam alto processamento de dados. O surgimento de processadores com vários núcleos de processamento é o maior fator para o crescimento substancial desta estratégia. Processamento de transações, recuperação de informação, mineração de dados e entre outras, são apenas alguns poucos exemplos de aplicações que podem se beneficiar deste poderoso artefato computacional [Grama *et al.*, 2003].

3.1 Computação Paralela

Computação paralela pode ser encarada como utilização concomitante de recursos computacionais, como processador, memória principal e memória secundária, para solucionar problemas que possam ser formulados como matemáticos computacionais. Esses recursos podem ser compreendidos como processadores com múltiplos cores ou até mesmo como um conjunto arbitrário de computadores interconectados.

Segundo Grama *et al*(2003) para desenvolvermos um algoritmo paralelo deve-se incluir todos ou apenas alguns dos seguintes tópicos:

- Identificar partes do trabalho que podem ser processadas simultaneamente.
- Atribuir cada parte do trabalho a diferentes processos.

- Distribuir os dados às tarefas.
- Gerenciar os dados compartilhados.
- Sincronizar os processos.

A decomposição é um processo que identifica e divide a computação em porções menores para o processamento simultâneo, como apresentado na Figura 3.1. Um dos principais desafios enfrentados na programação paralela é a decomposição do problema [Gropp *et al.*, 2003]. Na computação paralela, *tasks* são as partes definidas pela decomposição, sendo que podem possuir diversos tamanhos. Geralmente algumas tasks dependem dos resultados gerados por outras, causando dependência na solução, este fato pode ser representado pelo grafo de dependência, onde a ordem e as dependências são reproduzidas.

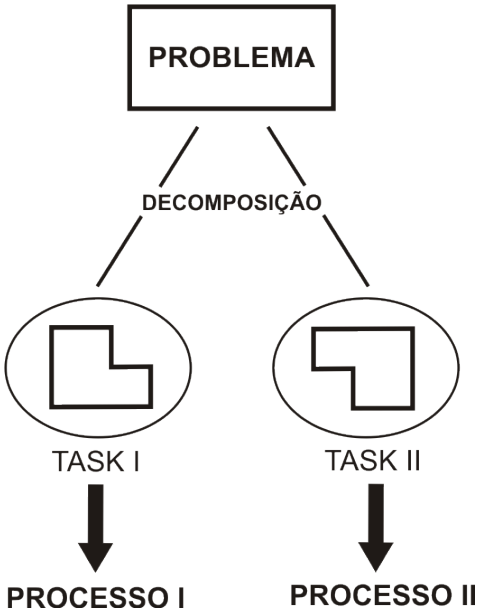

Figura 3.1: Ilustração do processo de decomposição de problemas.

O tamanho e a quantidade das *tasks* definem a granularidade da decomposição feita, ou seja, elas podem possuir granularidade grossa ou fina. No geral, quanto mais fina a granularidade, maior será o nível de concorrência, que é o número de tasks que são processadas em determinado tempo. Mas existe um limite para a granularidade, como por exemplo, o problema de multiplicação de matrizes, onde a estratégia paralela não pode possuir complexidade maior que n^2, pois em sua abordagem sequencial são feitas n^2 multiplicações e adições, como mostra a Figura 3.2.

Segundo Gropp *et al* (2003), existem duas estratégias de decomposição, a primeira é denominada de paralelismo de *tasks*, onde os principais cálculos do algoritmo e suas dependências são identificados, para que possam ser paralelizados. A segunda estratégia é conhecida como

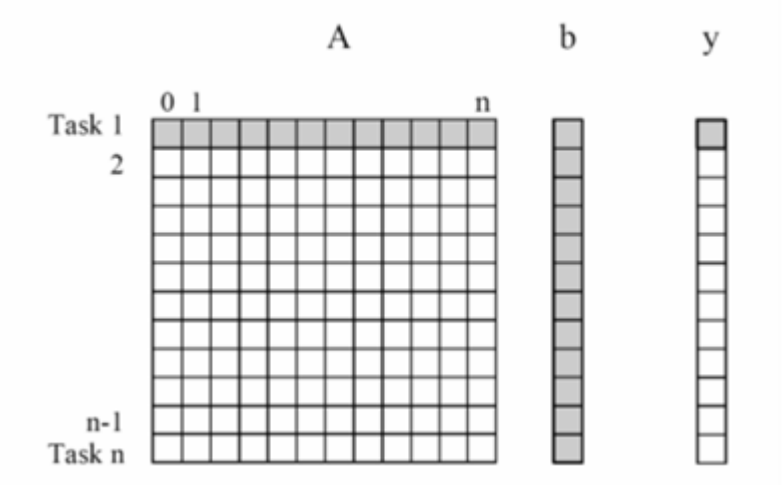

Figura 3.2: Problema de multiplicação de matrizes decomposto em *tasks* de granularidade fina. Figura retirada de Grama *et al.* (2003).

paralelismo de dados, pois ela subdivide um conjunto de dados de um determinado problema a qual se deseja paralelizar, e as atribui a processos diferentes.

Na maioria dos casos, os algoritmos paralelos apresentam uma maior carga computacional, devido à computação extra, comunicação e sincronização que estes são forçados a praticar. Desta forma, a melhor maneira de avaliar o desempenho dos métodos em paralelo é calcular o $SpeedUp(S_p)$, que é a razão entre o tempo T_1 (melhor método sequencial utilizando 1 (um) processo) gasto pelo tempo T_2 (método utilizando P processos), como mostra a equação abaixo:

$$Sp = \frac{T_1}{T_2} \tag{3.1}$$

A medida de eficiência (E) serve para mensurar a fração de tempo a qual o processo realmente foi utilizado, como podemos observar na equação (3.2). O ideal para aplicações paralelas seria que o $SpeedUp$ fosse igual ao número de processos, mas sabemos que na prática esta situação dificilmente acontece, pois geralmente o $SpeedUp$ é menor que o número de processos, assim o resultado dessa medida fica entre o intervalo $[0, 1]$.

$$E = \frac{SpeedUp}{NmeroDeProcessos} \tag{3.2}$$

3.2 Estratégias de Paralelização e o Modelo de ilhas

O algoritmo de Evolução Diferencial possui dificuldades de se chegar à convergência quando imposto a problemas de maior complexidade, ou seja, em funções-objetivo mais complexas, tornando o método lento [Tomassini, 1999]. Assim esta seção do texto apresenta 3 (três)

propostas de paralelização do algoritmo de Evolução Diferencial, sendo detalhada a abordagem de modelos de ilhas a qual é foco deste trabalho.

O algoritmo de Evolução Diferencial pode ser facilmente paralelizado principalmente pelo fato de cada indivíduo da sua população ser avaliado individualmente [Schwefel, 1995]. Sendo que, o único momento em que método necessita de outros indivíduos da população é para o processo de mutação diferencial [Tasoulis *et al.*, 2004].

Assim, a primeira estratégia emprega a abordagem de paralelismo de *tasks*, onde existe uma população única e as tarefas de mutação e avaliação de fitness (seleção) são distribuídas por um processo principal, denominado *master*, a outros processos chamados de *slaves*, que são responsáveis pelos cálculos. A Figura 3.3 apresenta como funciona sua topologia, que também é conhecida como *Master/Slave*.

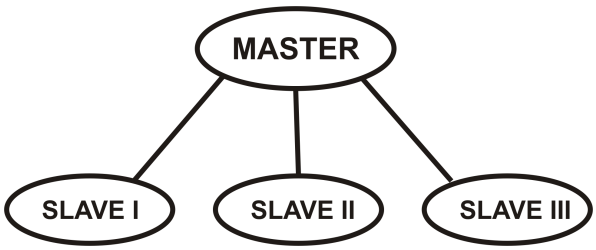

Figura 3.3: Topologia da estratégia de paralelização *Master/Slave*.

Entretanto, existem outras estratégias que empregam a abordagem de paralelismo de dados, nas quais estão os modelos de populações única e múltiplas, também conhecida como modelo de ilhas. O modelo de população única é uma abordagem de granularidade fina, onde consiste em uma população totalmente distribuída em muitos processos, na qual se comunicam com uma pequena vizinhança apenas para o processo de mutação e seleção, como mostra a Figura 3.4. Esta estratégia é um tanto problemática e não aconselhável quando o número de processos a disposição são limitados, ou quando estratégias de mutação que demandam informação de toda população são aplicadas, como as *best* e *current-to-best* já mencionadas anteriormente [Tasoulis *et al.*, 2004].

A terceira estratégia de paralelização é o modelo em ilhas, onde a população total é dividida em subpopulações ou também chamadas de demes, e em cada subpopulação o método de evolução diferencial é aplicado, como demonstrado na Figura 3.5. Esta abordagem possui granularidade grossa, diferentemente da segunda aqui apresentada, sendo cada subpopulação atribuída a um processo diferente.

Nesta estratégia, são gerados indivíduos aleatórios para cada uma das subpopulações que evoluem independentemente, permitindo que elas desenvolvam sua própria solução para o problema de otimização proposto. Desta maneira, as demes estão relativamente isoladas umas das outras, o que ajudar a manter uma maior diversidade genética, com isso, explorando diferentes partes do espaço de busca [Adamidis, 1998].

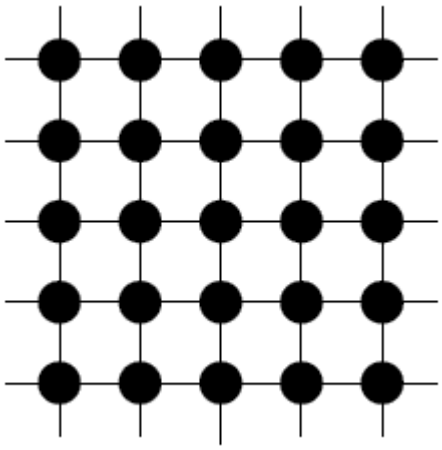

Figura 3.4: Abordagem de população única, população espacialmente distribuída.

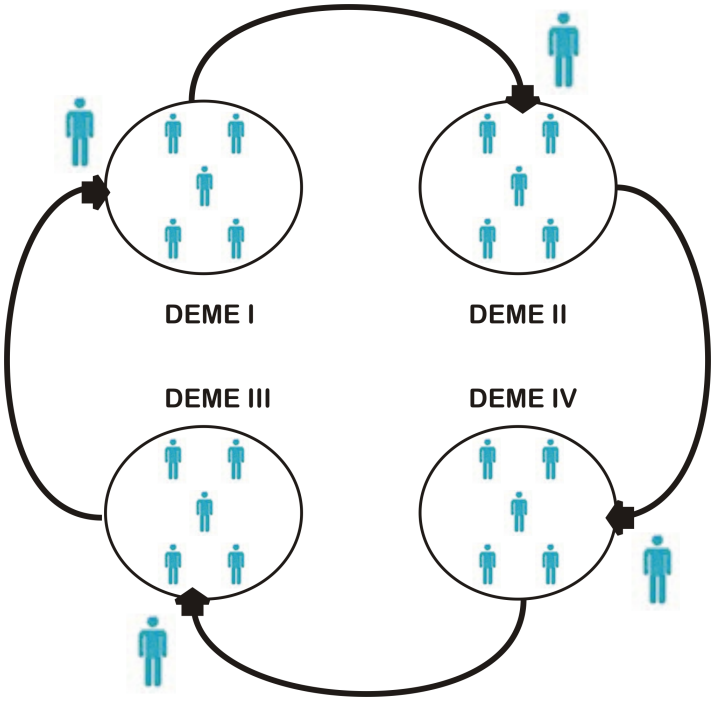

Figura 3.5: Estrutura exemplo do modelo de ilhas.

O processo de migração, ou seja, troca de região por um ou vários indivíduos é também proposta a esta estratégia com o intuito de que haja troca de informação entre as subpopulações, transmitindo dados sobre suas soluções encontradas, causando uma cooperação entre elas, de forma que a evolução de uma, auxilie a evolução da outra. Apresentamos assim na Figura 3.6, este processo.

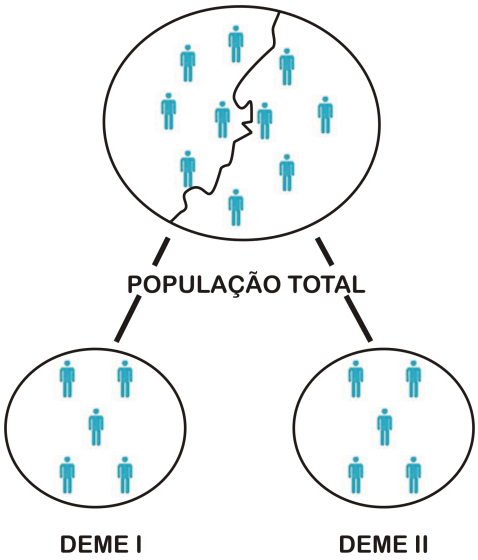

Figura 3.6: Decomposição da população em duas.

O modelo de ilhas, desperta o interesse dos pesquisadores, pelo fato de que se espera, com a utilização dessa estratégia de paralelização, uma diminuição no número de gerações que o método demora a alcançar para encontrar uma boa solução. E como em toda estratégia de paralelização deseja-se um melhor desempenho do método comparado com sua abordagem sequencial.

Existem propostos na literatura diversas abordagens para o processo de migração de indivíduos e topologia para representar o fluxo migratório das demes. Algumas destas abordagens são discutidas a seguir.

3.2.1 Política de migração

Em geral, as propostas mais conhecidas para esse processo, são:

- Um bom migrante substitui o pior indivíduo da população destino.
- Um bom migrante substitui um indivíduo escolhido aleatoriamente pela população destino.
- Um migrante escolhido aleatoriamente substitui o pior indivíduo da população destino.

- Um migrante escolhido aleatoriamente substitui um indivíduo escolhido aleatoriamente pela população destino.

Estudos realizados por Cantú-Paz (2001) demonstram que as escolhas da política e da taxa de migração possuem grande impacto na velocidade de convergência, fenômeno causado pela pressão seletiva que estas escolhas implicam.

Desta forma, podemos acompanhar este estudo visualizando as Figuras 3.7, 3.8, 3.9 e 3.10, onde verificamos que o tempo de *takeover* e números de gerações que algoritmo demora a convergir, são diretamente influenciados por taxas de migrações mais elevadas e pelas políticas que exercem maior pressão seletiva, como as políticas de migração que selecionam o melhor migrante[Cantú-Paz, 2001]. Trabalhos nesta área foram iniciados por Goldberg e Deb (1991), onde o conceito tempo de *takeover* foi introduzido, que significa o tempo que boas soluções demoram a dominar uma população.

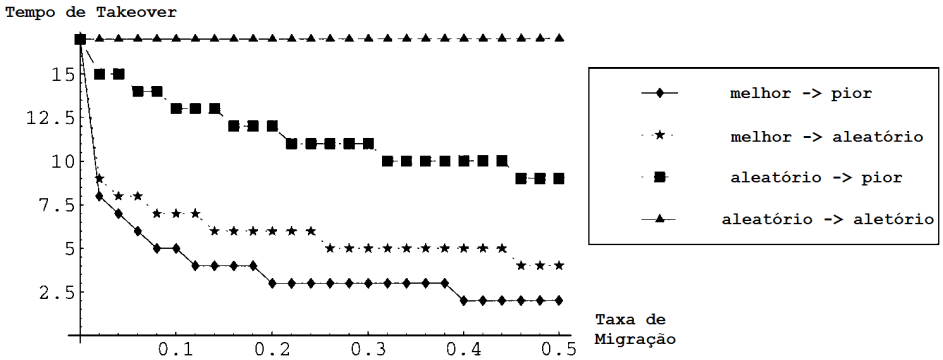

Figura 3.7: Relação tempo de takeover e taxa de migração, dada a política de migração. Figura traduzida de Cantú-Paz (2001).

Taxas de migração são valores compreendidos no intervalo $[0, 1]$ que determinam a porcentagem de migrações de uma subpopulação para outra, que irão ocorrer durante o processo de otimização. Outra forma de controlar a quantidade de migrações são os intervalos de migração, que são os intervalos que o método passará sem migrar nenhum indivíduo.

No entanto, não é recomendável uma pressão seletiva muito alta, pois pode fazer com que o método convirja muito rapidamente, ocasionando uma convergência prematura, fato esse que é não desejável para algoritmos evolutivos. Assim, a melhor maneira de livrar deste problema é escolher políticas de migração que causem menos pressão seletiva ou empregar menores taxas de migração ou intervalos de migração.

3.2.2 Topologia do fluxo de migratório

Sabemos até então que, a troca de informações entre as subpopulações contribui para a evolução das mesmas, e que o isolamento ou a comunicação total entre elas, são pontos críticos para custo computacional do modelo em ilhas.

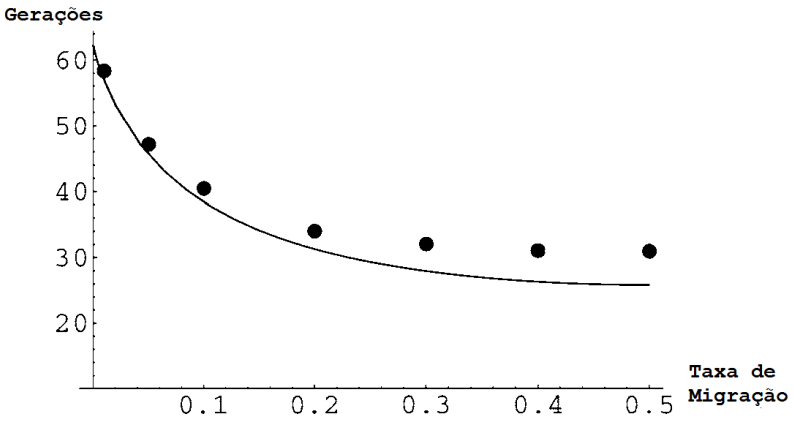

Figura 3.8: Relação número de gerações e taxa de migração, migrando o melhor indivíduo e substituindo pelo pior indivíduo da população destino. Figura traduzida de Cantú-Paz (2001).

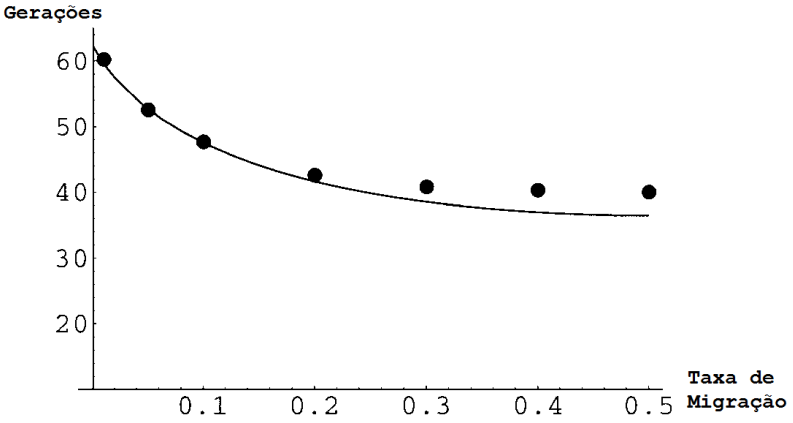

Figura 3.9: Relação número de gerações e taxa de migração, migrando o melhor indivíduo e substituindo por um indivíduo escolhido aleatoriamente pela população destino. Figura traduzida de Cantú-Paz (2001).

Contudo, o objetivo das topologias dos fluxos migratórios é oferecer estruturas de comunicação estáticas que garantam bom desempenho ao método, sem que sofram dos problemas de isolamento ou comunicação excessiva das subpopulações. De modo geral, as topologias mais conhecidas para o modelo de ilhas são: anel e escada.

Existem diversas variações destas topologias, mas apresentamos na Figura 3.11 as topologias em anel e escada bidirecionais.

3.2.3 Abordagem proposta

Este trabalho propõe, uma implementação paralela do algoritmo de Evolução Diferencial Autoadaptativo, empregando o modelo em ilhas visando uma diminuição do número de gerações

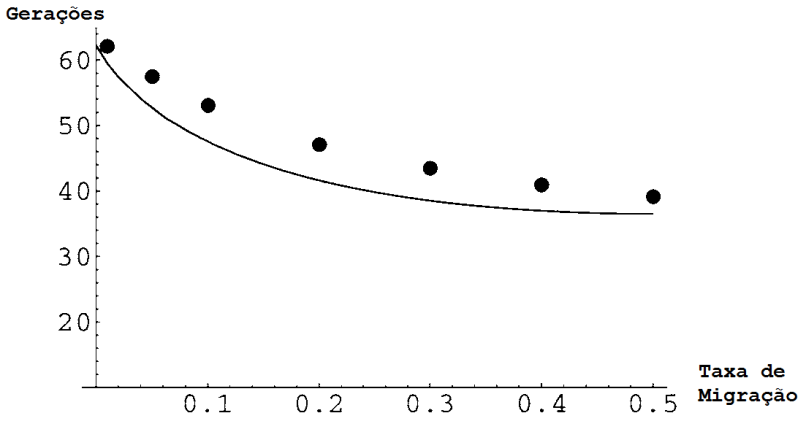

Figura 3.10: Relação número de gerações e taxa de migração, migrando um indivíduo escolhido aleatoriamente e substituindo por um indivíduo escolhido aleatoriamente pela população destino. Figura traduzida de Cantú-Paz (2001).

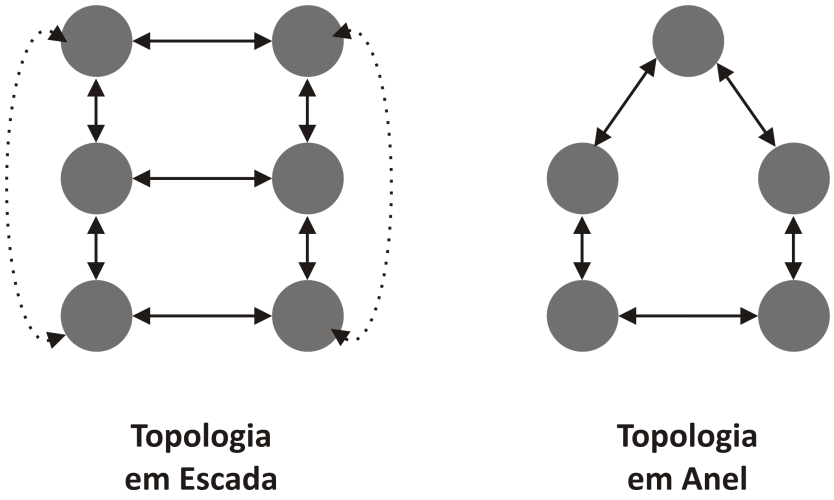

Figura 3.11: Exemplos de tologias em escada e anel bidirecionais.

que o método em sequencial gasta, consequentemente gerando uma melhora no desempenho do mesmo, além de fazer uma busca mais eficiente [Alba and Tomassini, 2002].

Apesar de o método exigir muitos cálculos para se chegar a uma boa solução, é possível acompanhar que este não é um trabalho muito pesado para uma abordagem sequencial, mas o fato deste método ter dificuldades para convergência com problemas mais complexos, nos leva a propor melhorias pra tal. Assim, uma abordagem paralela usando o modelo em ilhas é boa solução para melhoria, pois possui granularidade mais grossa e a divisão de sua população nos leva a uma melhor exploração do espaço de busca. Ao passo que, o uso de uma granularidade mais fina demandaria mais recursos computacionais, com isso o nível de concorrência aumentaria muito para os recursos disponíveis, o que poderia fazer com que seu desempenho não melhorasse.

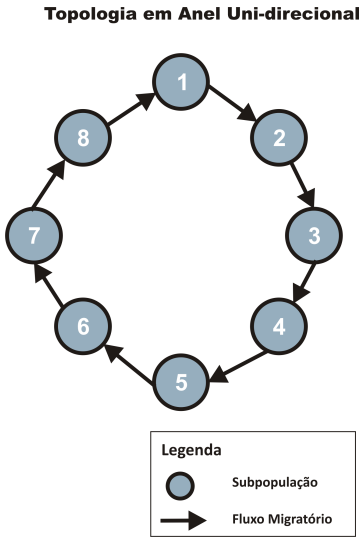

Figura 3.12: Exemplos de tologias em escada e anel bidirecionais.

Contudo, propomos que seja desenvolvido um modelo em ilhas empregando a topologia em anel unidirecional, onde em cada subpopulação existe um algoritmo de Evolução Diferencial Autoadaptativo, como apresentado na Figura 3.12. Sua implementação foi desenvolvida multi-thread usando a linguagem de programação Java, onde existe um espaço de memória compartilhada pelas mesmas para que os indivíduos que serão migrados sejam compartilhados por tais. A thread principal é responsável por repassar os migrantes oriundos das subpopulações às demes destino, como podemos observar no pseudocódigo abaixo.

Algoritmo do Modelo em Ilhas

Thread Principal
for $i = 1$ to $NumeroDeSubpopulacoes$ **do**
  $thread(i) \leftarrow supopulacao(i)$
end for
for $i = 1$ to 10 **do**
  Receba um individuo de cada subpopulacao
  Envie o indivíduo recebido a cada subpopulacao
  if $CriterioDeParadaAtendido$ **then**
    Envie sinal de termino a todas subpopulacoes
  end if
end for
Em cada subpopulacao
for $i = 1$ to $NumerodeGeracoes$ **do**
  for $i = 1$ to 10 **do**
    Execute o algoritmo de evolucao diferencial

```
    end for
    Envio o migrante para a Thread principal
    if RecebeuMigrante then
      Substitua-o na populacao
    end if
    if RecebeuSinalDeTermino then
      Finaliza a execucao
    end if
  end for
```

Neste trabalho, as políticas de migração aplicadas serão um bom migrante substitui um indivíduo escolhido aleatoriamente pela população destino e um migrante escolhido aleatoriamente substitui um indivíduo também escolhido aleatoriamente pela população destino. Desta forma, podemos avaliar como a seleção do melhor migrante pode influenciar ou não no desempenho do método.

Para que o método não sofra de problemas como muita sincronização das threads, que diminuiria o desempenho, ou de pressão seletiva muito alta, que poderia levar a convergência prematura, é utilizado um intervalo de migração de 10 (dez) gerações. Espera-se que com este intervalo de migração, o algoritmo aprenda mais sobre os parâmetros que está buscando adaptar e faça também melhores buscas no espaço de soluções antes que a troca de informações com outra população ocorra.

Capítulo 4

Apresentação e Análise dos Resultados

Este capítulo apresenta os resultados alcançados para os diferentes testes realizados nesse trabalho, onde várias funções-objetivo foram testadas, assim como diversas configurações da abordagem paralela do algoritmo de Evolução Diferencial Autoadaptativa.

4.1 Configurações dos Testes

Um esquema de teste foi preparado para que cada problema de otimização fosse avaliado empregando as duas políticas de migração propostas anteriormente, juntamente com variações das configurações das ilhas, como podemos observar no esquema representado na Figura 4.1.

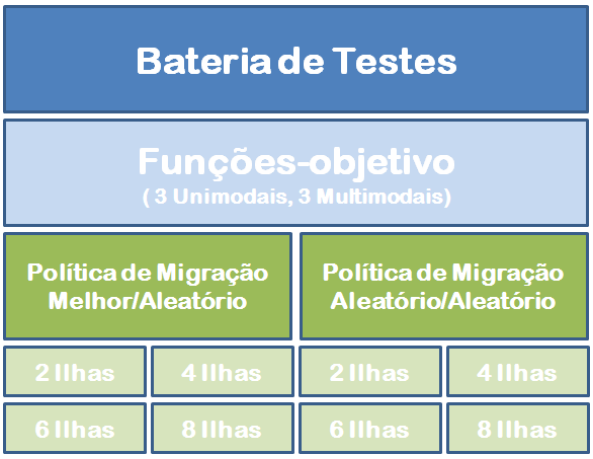

Figura 4.1: Esquema que estrutura a organização dos experimentos para avaliação do trabalho.

Em um computador com processador AMD Athlon II Dual-Core M300 2.0GHz e 2GB de memória RAM, cada função foi executada 30 vezes de forma independente e sua média aritmética considerada como resultado final ao experimento.

Sendo assim, descrevemos a seguir as funções-objetivo e as configurações da abordagem paralela utilizadas para realização dos experimentos.

4.1.1 Funções de teste

Funções que apresentam características diferenciadas foram escolhidas com intuito de observar o desempenho do método paralelo em diversificados cenários. Desse modo, as seguintes funções-objetivo foram utilizadas:

- Função 1

$$f_1(x) = \sum_{i=1}^{D} x_i^2 \ , \quad min = 0.0, \quad x_i \in [-100, 100] \tag{4.1}$$

- Função 2

$$f_2(x) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i| \ , \quad min = 0.0, \quad x_i \in [-10, 10] \tag{4.2}$$

- Função 3

$$f_3(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2 \ , \quad min = 0.0, \quad x_i \in [-100, 100] \tag{4.3}$$

- Função 4

$$f_4(x) = \sum_{i=1}^{D} -x_i \sin(\sqrt{|x_i|}) \ , \quad min = -12569.5, \quad x_i \in [-500, -500] \tag{4.4}$$

- Função 5

$$f_5(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10) \ , \quad min = 0.0, \quad x_1 \in [-5.12, 5.12] \tag{4.5}$$

- Função 6

$$f_6(x) = \frac{1}{4000} \sum_{i=D}^{n} x_i^2 - \prod_{i=D}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \ , \quad min = 0.0, \quad x_i \in [-600, 600] \tag{4.6}$$

As funções de 1 a 3 são unimodais, possuem muitas dimensões e foi definida uma precisão de $1.e^{-10}$. Já as funções de 4 a 6 são multimodais e o número de mínimos locais aumentam de forma exponencial com o número de dimensões. Para as funções multimodais foi definida uma precisão de $1.e^{-6}$.

Na Tabela 4.1, especificamos as configurações de alguns parâmetros das funções de teste, onde D representa a dimensão do problema e NP o tamanho da população total a qual iremos utilizar.

Função de Teste	Dimensão (D)	Tamanho da População (NP)
Função 1	100	100
Função 2	100	100
Função 3	30	100
Função 4	30	100
Função 5	30	100
Função 6	30	100

Tabela 4.1: Parâmetros das Funções de Teste.

4.1.2 Configurações da abordagem paralela

Como proposto, para cada problema teste empregamos as políticas de migração, que migram os melhores ou indivíduos escolhidos aleatoriamente, por outros indivíduos também escolhidos aleatoriamente na população destino, na qual denominaremos respectivamente de *melhor/aleatório* e *aleatório/aleatório*, de forma a simplificar o modo de referenciar estas políticas.

Assim, para cada par de política de migração e função objetivo foram realizados 4 (quatro) experimentos, onde houve variações na divisão da população total aplicada na abordagem sequencial, com a intenção de verificar o comportamento do método paralelo quando o número de subpopulações cresce. A Tabela 4.2 apresenta as características das subpopulações para cada configuração.

Modelo em Ilhas	Características das subpopulações
Duas Ilhas	2 subpopulações com 50 indivíduos cada
Quatro Ilhas	4 subpopulações com 25 indivíduos cada
Seis Ilhas	5 subpopulações com 16 indivíduos cada e 1 subpopulação com 20 indivíduos
Oito Ilhas	7 subpopulações com 12 indivíduos cada e 1 subpopulação com 16 indivíduos

Tabela 4.2: Características das subpopulações para cada modelo em ilhas.

4.2 Resultados

Apresentamos a seguir os resultados alcançados por cada função objetivo proposta neste trabalho.

4.2.1 Função 1

A Figura 4.2 apresenta o número médio de gerações que a abordagem sequencial e as estratégias paralelas gastam para a convergência em cada um dos métodos. Comparando as políticas de migração aplicadas a este problema, verificamos que, ao migrarmos os melhores diminuímos o número de gerações. Dessa forma, as estratégias que empregam a política *melhor/aleatório*, apresentaram, no geral, melhores resultados que a abordagem sequencial para este problema proposto.

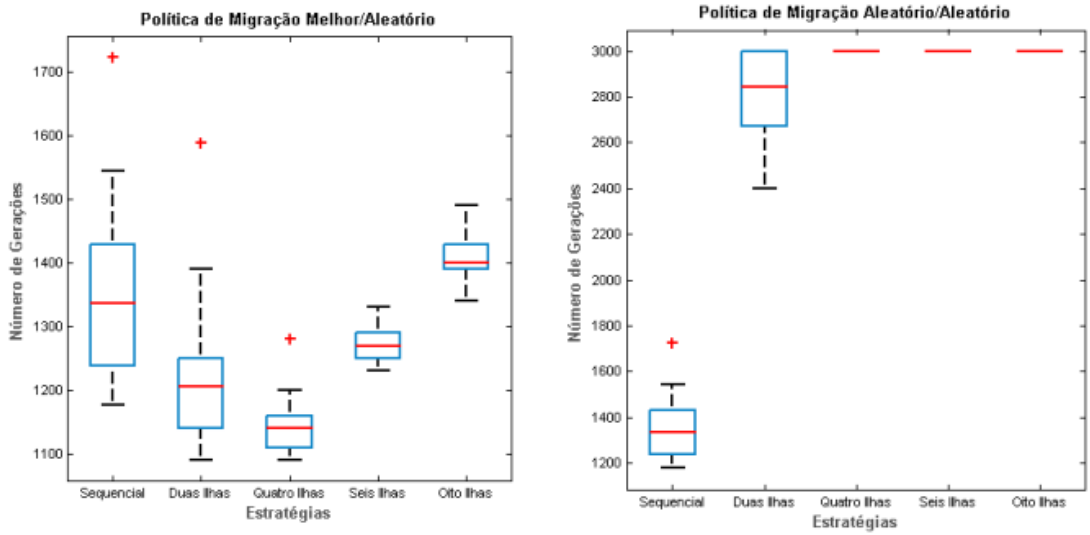

Figura 4.2: Gráficos que apresenta o número médio, mínimo e máximo de gerações gastos em cada método, até que o critério de parada seja atingido, para a Função 1.

Podemos verificar nas Tabelas 4.3 e 4.4 que o tempo de execução do método foi reduzido na maioria das estratégias que fizeram o uso da política que migra as melhores soluções.

	Sequencial	**Duas Ilhas**	**Quatro Ilhas**	**Seis Ilhas**	**Oito Ilhas**
Tempo(ms)	104.281	88.55	86.72	94.567	107.257

Tabela 4.3: Tempo de Execução para a Função 1, até que o critério de parada seja atingido - Política de Migração *Melhor/Aleatório*.

	Sequencial	**Duas Ilhas**	**Quatro Ilhas**	**Seis Ilhas**	**Oito Ilhas**
Tempo(ms)	104.281	217.411	226.665	226.3	225.928

Tabela 4.4: Tempo de Execução para a Função 1, até que o critério de parada seja atingido - Política de Migração *Aleatório/Aleatório*.

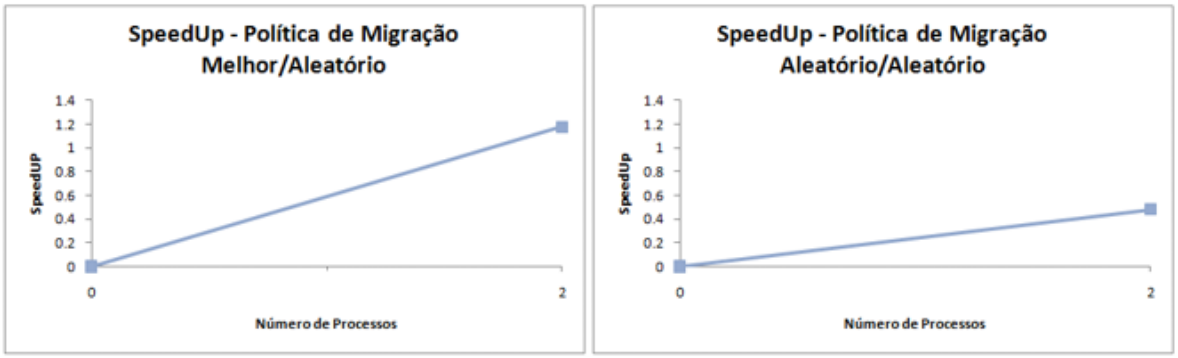

Figura 4.3: Gráficos que apresenta o *SpeedUp* para a estratégia de duas ilhas empregando as políticas de migração *Melhor/Aleatório* e *Aleatório/Aleatório* para a Função 1.

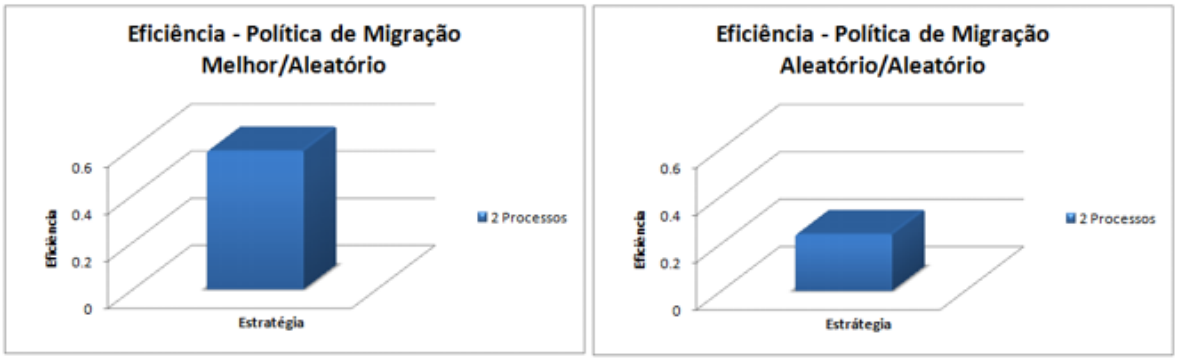

Figura 4.4: Gráficos que apresenta a eficiência para a estratégia de duas ilhas empregando as políticas de migração *Melhor/Aleatório* e *Aleatório/Aleatório* para a Função 1.

A Figura 4.3 representa o *SpeedUp* da estratégia de duas ilhas, de acordo com a política de migração empregada. Contudo observando-a, verificamos que a estratégia que utiliza a política *melhor/aleatório*, o *SpeedUp* alcançado é maior que 1 (um), pois, como já apresentado acima, os tempos de execução da desta configuração é menor que a abordagem sequencial, fato este que é desejável quando paralelizamos algum algoritmo.

O resultado da eficiência desta estratégia de duas ilhas pode ser observado na Figura 4.4. Assim, da mesma maneira que o *SpeedUp* da estratégia que emprega a política de migração *melhor/aleatório* foi melhor que a outra política, a eficiência seguiu a mesma tendência.

Assistindo a Figura 4.5 e as Tabelas 4.5 e 4.6 que detalham a melhor solução, a média e o desvio padrão cada estratégia, observamos que as estratégias paralelas encontraram melhores soluções que a abordagem sequencial.

Dadas as Tabelas 4.5 e 4.6, escolhemos as configurações paralelas que obtiveram o melhor valor médio das soluções encontradas para cada política de migração e comparamos sua curva de convergência com a da abordagem sequencial. Pode-se verificar que a abordagem sequencial e o modelo com 4 (quatro) ilhas, empregando a política de migração *melhor/aleatório*, chegam

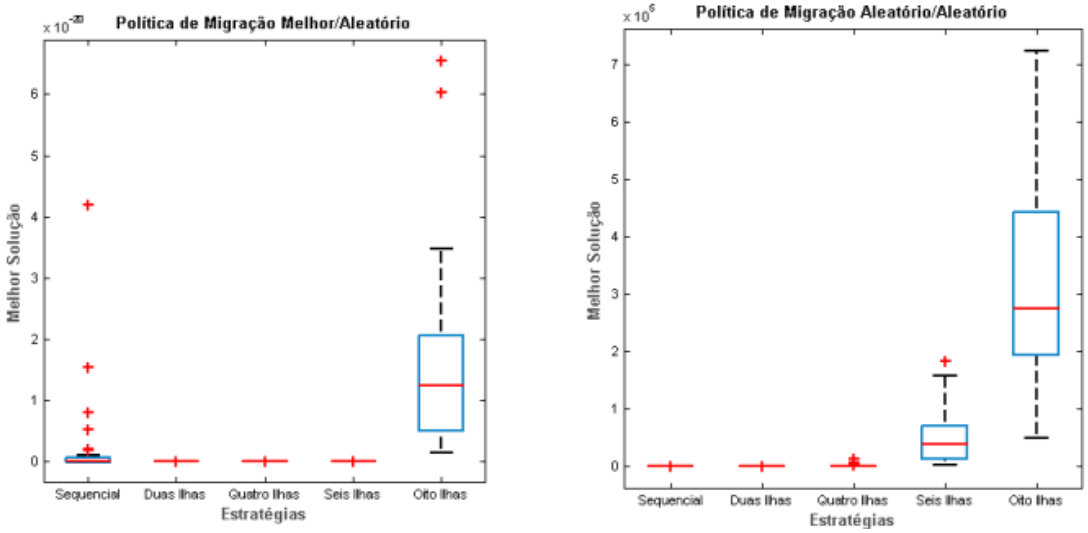

Figura 4.5: Gráficos que apresentam as melhores soluções encontradas para as variações do método, onde o critério de parada é atingir 3000 (três mil) gerações, para a Função 1.

	Sequencial	**Duas Ilhas**	**Quatro Ilhas**	**Seis Ilhas**	**Oito Ilhas**
Melhor	$1.23e^{-25}$	$4.9091e^{-28}$	$3.7922e^{-28}$	$5.8573e^{-25}$	$1.3892e^{-21}$
Média	$2.5737e^{-21}$	$1.7474e^{-25}$	$1.2608e^{-26}$	$9.259e^{-24}$	$1.6211e^{-20}$
Desvio Padrão	$8.0807e^{-21}$	$4.671e^{-25}$	$1.4413e^{-26}$	$1.3556e^{-23}$	$1.576e^{-20}$

Tabela 4.5: Melhor solução encontrada, média e desvio padrão encontrados, onde o critério de parada é atingir 3000 (três mil) gerações, para a Função 1 - Política de Migração *Melhor/Aleatório*.

	Sequencial	**Duas Ilhas**	**Quatro Ilhas**	**Seis Ilhas**	**Oito Ilhas**
Melhor	$8.3592e^{-25}$	$3.93e^{-09}$	19.7738	2477.4	50267
Média	$2.5737e^{-21}$	0.0013	1086.1	50779	311070
Desvio Padrão	$8.0807e^{-21}$	0.0041	2875.5	44441	162530

Tabela 4.6: Melhor solução encontrada, média e desvio padrão encontrados, onde o critério de parada é atingir 3000 (três mil) gerações, para a Função 1 - Política de Migração *Aleatório/Aleatório*.

mais próximos a um boa solução em menos gerações, como demonstra a Figura 4.6.

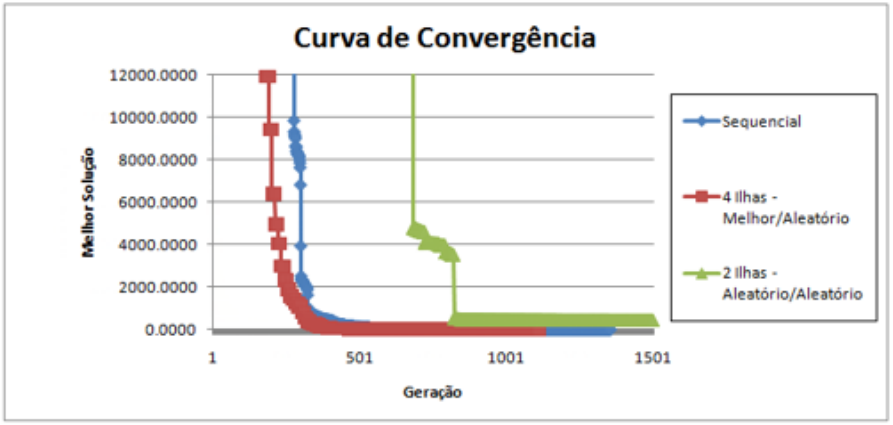

Figura 4.6: Gráficos que apresentam as curvas de convergência das abordagem paralelas e sequencial, para a Função 1.

Contudo, a estratégia paralela que utiliza quatros ilhas foi a que apresentou resultados melhores. O modelo com 4 (quatro) ilhas obteve vantagens em vários aspectos como: menor número médio de gerações gastos para convergir, menor tempo de execução e melhores soluções dado um número fixo de gerações.

4.2.2 Função 2

Para este problema teste, todas as estratégias de paralelização que empregam a política de migração *melhor/aleatório* diminuíram o número de gerações em comparação com a versão sequencial. No entanto, as estratégias que empregam a outra política não conseguiram convergir em nenhum dos casos, como podemos observar na Figura 4.7.

Como podemos observar nas Tabelas 4.7 e 4.8, o tempo computacional gasto por algumas estratégias da abordagem paralela diminuíram. Este processo é explicado principalmente pelo fato que essas mesmas estratégias tiveram o número de gerações reduzido, assim, como consequência o tempo computacional decresce.

	Sequencial	**Duas Ilhas**	**Quatro Ilhas**	**Seis Ilhas**	**Oito Ilhas**
Tempo(ms)	164.129	120.804	74.775	71.069	74.734

Tabela 4.7: Tempo de Execução para a Função 2, até que o critério de parada seja atingido - Política de Migração *Melhor/Aleatório*.

Assim como ocorreu com o tempo computacional e o número de gerações, as medidas de *SpeedUp* e eficiência, tiveram bons resultados devido à diferença entre os tempos da abor-

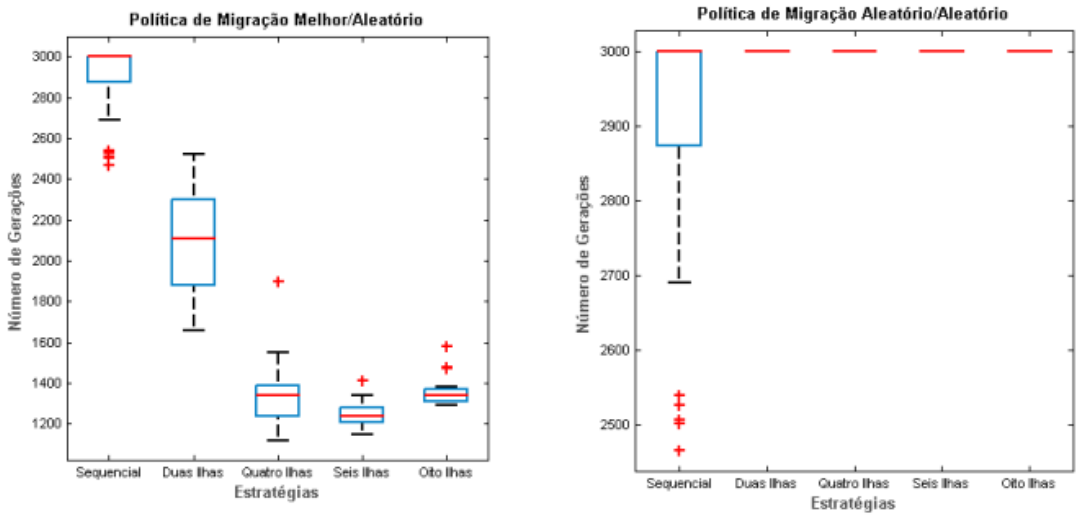

Figura 4.7: Gráficos que apresenta o número médio, mínimo e máximo de gerações gastos em cada método, até que o critério de parada seja atingido, para a Função 2.

	Sequencial	**Duas Ilhas**	**Quatro Ilhas**	**Seis Ilhas**	**Oito Ilhas**
Tempo(ms)	164.129	170.966	166.613	165.182	163.42

Tabela 4.8: Tempo de Execução para a Função 2, até que o critério de parada seja atingido - Política de Migração *Aleatório/Aleatório*.

dagem sequencial e a paralela, já verificados anteriormente. Este fato está representado nas Figuras 4.8 e 4.9.

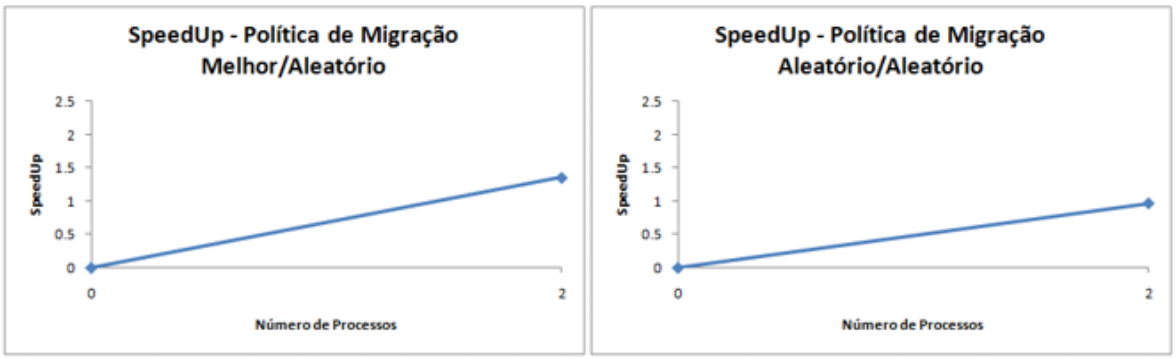

Figura 4.8: Gráficos que apresenta o *SpeedUp* para a estratégia de duas ilhas empregando as políticas de migração *Melhor/Aleatório* e *Aleatório/Aleatório* para a Função 2.

A eficácia da abordagem paralela usando a política de migração melhor/aleatório pode ser comprovada através da Tabela 4.9 e a Figura 4.10. Nessas verificamos que o modelo de seis ilhas é a estratégia que possui o menor valor médio para as soluções encontradas, assim como,

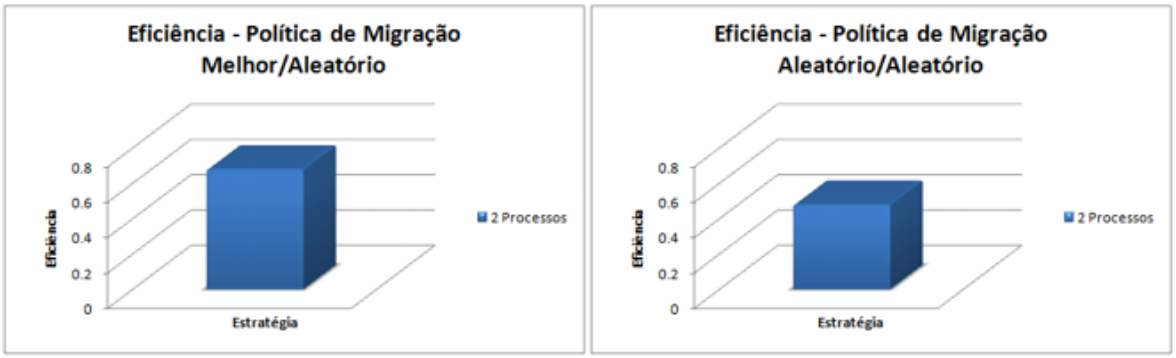

Figura 4.9: Gráficos que apresenta a eficiência para a estratégia de duas ilhas empregando as políticas de migração *melhor/aleatório* e *aleatório/aleatório* para a Função 2.

a melhor de todas as soluções encontradas para o problema teste.

	Sequencial	**Duas Ilhas**	**Quatro Ilhas**	**Seis Ilhas**	**Oito Ilhas**
Melhor	$1.11e^{-08}$	$6.76e^{-13}$	$2.22e^{-19}$	$1.91e^{-19}$	$5.68e^{-18}$
Média	0.0143	$1.23e^{-07}$	$5.95E^{-}17$	$1.79E^{-}17$	$3.87E^{-}17$
Desvio Padrão	0.0579	$6.35e^{-07}$	$1.49e^{-16}$	$6.08e^{-17}$	$2.49e^{-17}$

Tabela 4.9: Melhor solução encontrada, média e desvio padrão encontrados, onde o critério de parada é atingir 3000 (três mil) gerações, para a Função 1 - Política de Migração *Melhor/Aleatório*.

No entanto, as estratégias que utilizaram a política de migração *aleatório/aleatório* não conseguiram encontrar resultados superiores aos resultados da abordagem sequencial, como podemos acompanhar na Tabela 4.10 e Figura 4.10.

	Sequencial	**Duas Ilhas**	**Quatro Ilhas**	**Seis Ilhas**	**Oito Ilhas**
Melhor	$8.3592e^{-25}$	$3.93e^{-09}$	19.7738	2477.4	50267
Média	$2.5737e^{-21}$	0.0013	1086.1	50779	311070
Desvio Padrão	$8.0807e^{-21}$	0.0041	2875.5	44441	162530

Tabela 4.10: Melhor solução encontrada, média e desvio padrão encontrados, onde o critério de parada é atingir 3000 (três mil) gerações, para a Função 2 - Política de Migração *Aleatório/Aleatório*.

Seguindo os mesmos dados apresentados nas tabelas anteriores, a Figura 4.10 demonstra em forma de gráficos os resultados já discutidos.

Escolhendo as configurações paralelas que tiveram melhor desempenho, acerca do número médio de gerações, podemos notar através do gráfico que representa a curva de convergência das abordagens que o modelo de 6 (seis) ilhas o qual emprega a política de migração

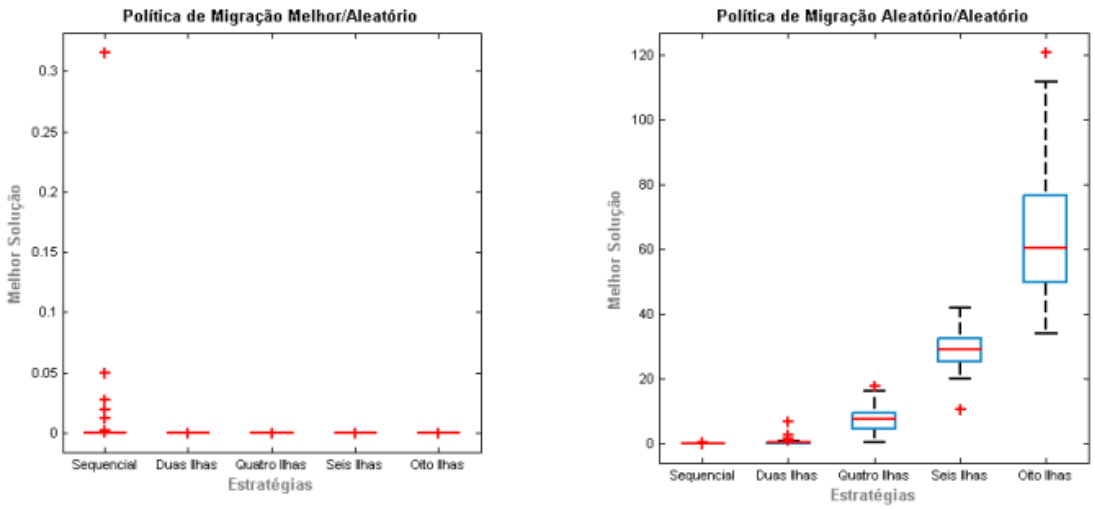

Figura 4.10: Gráficos que apresentam as melhores soluções encontradas para as variações do método, onde o critério de parada é atingir 3000 (três mil) gerações, para a Função 2.

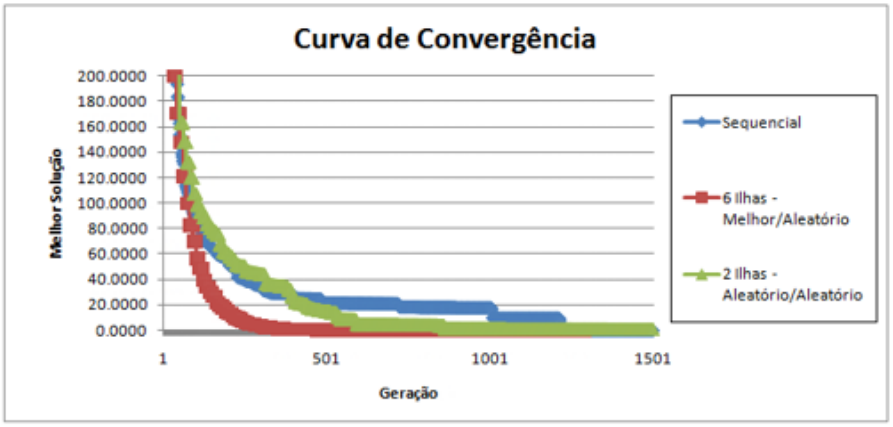

Figura 4.11: Gráficos que apresentam as curvas de convergência das abordagem paralelas e sequencial, para a Função 2.

melhor/aleatório converge para o mínimo mais rápido que as outras abordagens, como apresentado na Figura 4.11.

Apesar de a outra configuração paralela convergir para um valor próximo ao mínimo mais rápido que a abordagem sequencial, esta fica pressa nestes valores sem que consiga encontrar uma boa solução, ou seja, esta configuração converge prematuramente.

Para a função 2, o modelo de quatros e seis ilhas demonstraram ótimos resultados comparados à abordagem sequencial e entre as estratégias paralelas. Estes resultados foram garantidos pelo fato do método convergir mais rápido para uma boa solução, gastando menos gerações, desta maneira, possuíam mais gerações para que encontrassem soluções melhores do que as já

obtidas até o momento.

4.2.3 Função 3

Dada esta função de teste, o número médio de gerações que as estratégias paralelas atingiram nos experimentos, foram em todos os casos maiores que os resultados obtidos pela abordagem sequencial, como representado na Figura 4.12.

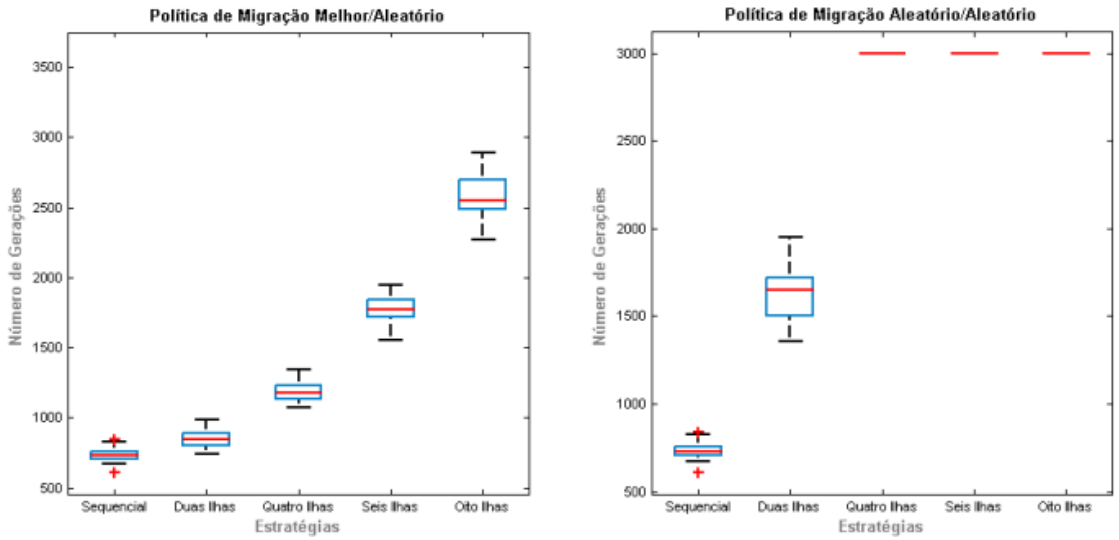

Figura 4.12: Gráficos que apresenta o número médio, mínimo e máximo de gerações gastos em cada método, até que o critério de parada seja atingido, para a Função 3.

Verificamos analisando as Tabelas 4.11 e 4.12 que os tempos de execução atingidos pelas configurações paralelas foram todos maiores que na abordagem sequencial, fato explicado pelo maior número médio de gerações dos modelos em paralelo, para determinar uma boa solução.

	Sequencial	**Duas Ilhas**	**Quatro Ilhas**	**Seis Ilhas**	**Oito Ilhas**
Tempo(ms)	21.01	47.273	87.224	88.308	86.791

Tabela 4.11: Tempo de Execução para a Função 3, até que o critério de parada seja atingido - Política de Migração *Melhor/Aleatório*.

	Sequencial	**Duas Ilhas**	**Quatro Ilhas**	**Seis Ilhas**	**Oito Ilhas**
Tempo(ms)	21.01	47.273	87.224	88.308	86.791

Tabela 4.12: Tempo de Execução para a Função 3, até que o critério de parada seja atingido - Política de Migração *Aleatório/Aleatório*.

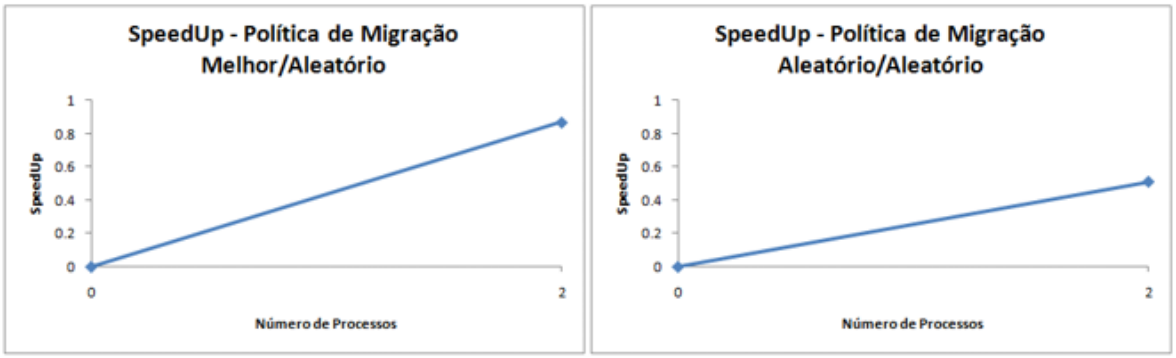

Figura 4.13: Gráficos que apresenta o *SpeedUp* para a estratégia de duas ilhas empregando as políticas de migração *Melhor/Aleatório* e *Aleatório/Aleatório* para a Função 3.

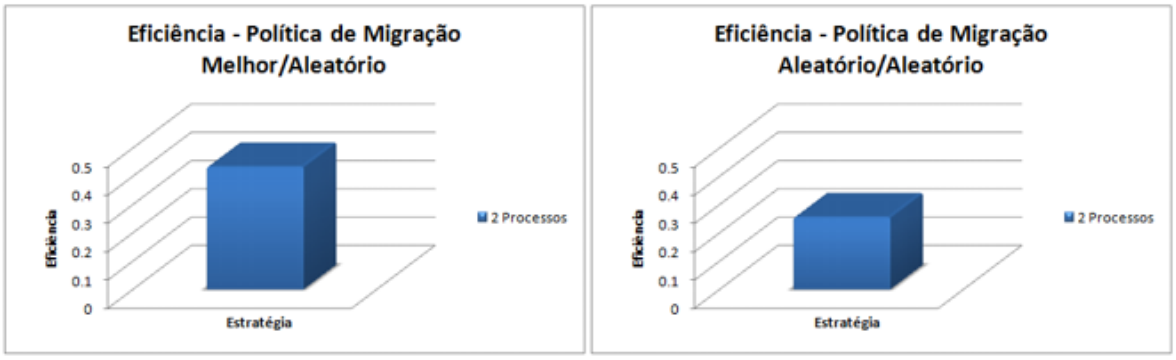

Figura 4.14: Gráficos que apresenta a eficiência para a estratégia de duas ilhas empregando as políticas de migração *Melhor/Aleatório* e *Aleatório/Aleatório* para a Função 3.

O *SpeedUp* e a eficiência calculados para os resultados dos experimentos, aplicando este problema teste, não foram bons, como explicado nas Figuras 4.13 e 4.14. Sabemos, entretanto que, os resultados das abordagens sequencial e paralela possuem influência sobre esta medida e, como já analisados anteriormente, as abordagens paralelas alcançaram resultados aquém do esperado.

Apesar das configurações paralelas terem tido resultados abaixo do esperado, suas soluções encontradas, quando submetidas a um fixo de gerações, são em sua maioria boas soluções para o problema em questão, como explicado nas Tabelas 4.13 e 4.14.

Como explicado subsequente e verificado na Figura 4.15, para as várias configurações da abordagem paralela as melhores soluções estão muito próximas do mínimo, que neste caso é 0 (zero).

É perceptível na Figura 4.16, que a abordagem sequencial encontra uma boa solução antes que as configurações paralelas, o que pode ser demonstrado pela sua curva de convergência. A versão não paralelizada é seguida pelo modelo de 2 (duas) ilhas que emprega a política de migração *melhor/aleatório*.

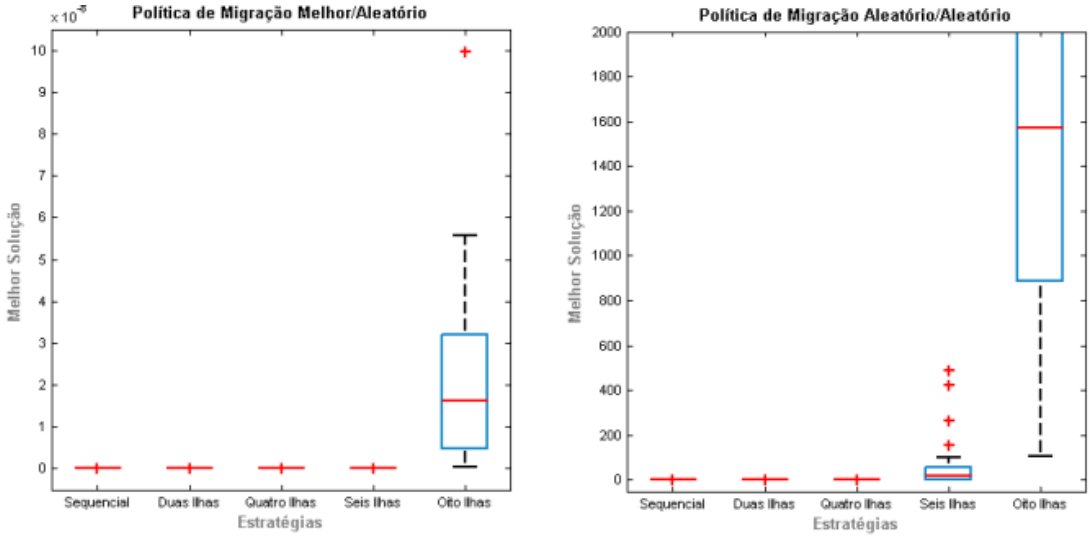

Figura 4.15: Gráficos que apresentam as melhores soluções encontradas para as variações do método, onde o critério de parada é atingir 3000 (três mil) gerações, para a Função 3.

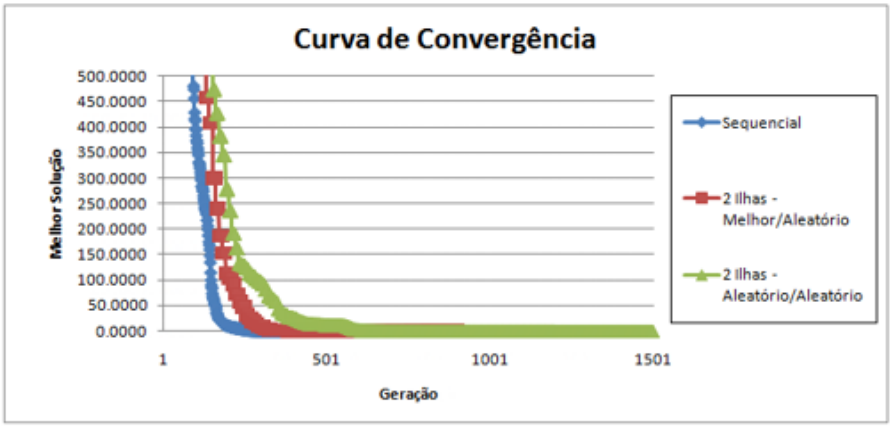

Figura 4.16: Gráficos que apresentam as curvas de convergência das abordagem paralelas e sequencial, para a Função 3.

	Sequencial	Duas Ilhas	Quatro Ilhas	Seis Ilhas	Oito Ilhas
Melhor	$7.722e^{-42}$	$4.7324e^{-35}$	$8.4959e^{-23}$	$3.56e^{-15}$	$3.77e^{-10}$
Média	$1.211e^{-36}$	$1.6641e^{-31}$	$1.6561e^{-20}$	$8.73e^{-13}$	$2.28e^{-08}$
Desvio Padrão	$3.0157e^{-36}$	$4.3321e^{-31}$	$2.7168e^{-20}$	$1.31e^{-12}$	$2.23e^{-08}$

Tabela 4.13: Melhor solução encontrada, média e desvio padrão encontrados, onde o critério de parada é atingir 3000 (três mil) gerações, para a Função 3 - Política de Migração *Melhor/Aleatório*.

	Sequencial	Duas Ilhas	Quatro Ilhas	Seis Ilhas	Oito Ilhas
Melhor	$7.7224e^{-42}$	$1.7012e^{-17}$	0.00056264	0.4349	107.9449
Média	$1.211e^{-36}$	$1.09e^{-13}$	0.0362	70.2873	1731.5
Desvio Padrão	$3.0157e^{-36}$	$3.95e^{-13}$	0.0611	120.8642	1291.8

Tabela 4.14: Melhor solução encontrada, média e desvio padrão encontrados, onde o critério de parada é atingir 3000 (três mil) gerações, para a Função 3 - Política de Migração *Aleatório/Aleatório*.

Desta maneira, verificamos que para esta função objetivo, o método em paralelo não obteve resultados melhores que a abordagem em comparação. Mas apesar de tudo, as soluções encontradas principalmente pelo modelo de duas ilhas estão muito próximas ao mínimo do problema em questão.

4.2.4 Função 4

Diante da função de teste 4 é possível notar, através da Figura 4.17, que nenhuma das abordagens do método descritas neste trabalho foram capazes de convergir, ou seja, encontrar um valor próximo ao mínimo global antes que o critério de parada fosse atendido, onde neste problema foi estimado 5000 (cinco mil) gerações.

Além disso, as Tabelas 4.15 e 4.16 demonstram que as estratégias de paralelização não obtiveram nenhum ganho significativo de tempo computacional em relação a abordagem sequencial.

	Sequencial	Duas Ilhas	Quatro Ilhas	Seis Ilhas	Oito Ilhas
Tempo(ms)	164.05	162.599	163.66	164.611	166.546

Tabela 4.15: Tempo de Execução para a Função 4, até que o critério de parada seja atingido - Política de Migração *Melhor/Aleatório*.

O *SpeedUp* calculado para os resultados dos tempos computacionais alcançados dessas estratégias estão próximos a 1 (um), como apresenta a Figura 4.18.

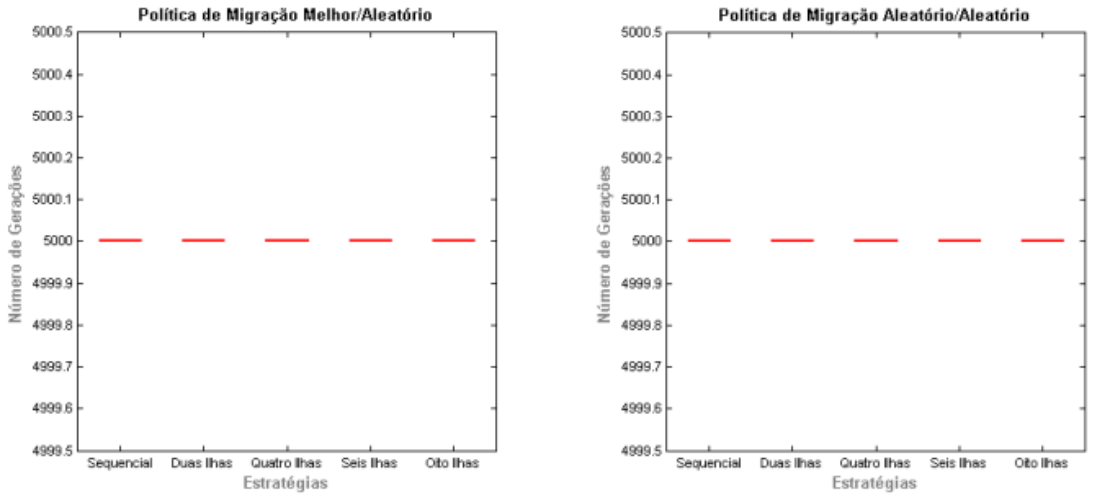

Figura 4.17: Gráficos que apresenta o número médio, mínimo e máximo de gerações gastos em cada método, até que o critério de parada seja atingido, para a Função 4.

	Sequencial	**Duas Ilhas**	**Quatro Ilhas**	**Seis Ilhas**	**Oito Ilhas**
Tempo(ms)	164.05	162.709	164.596	165.485	165.766

Tabela 4.16: Tempo de Execução para a Função 4, até que o critério de parada seja atingido - Política de Migração *Aleatório/Aleatório*.

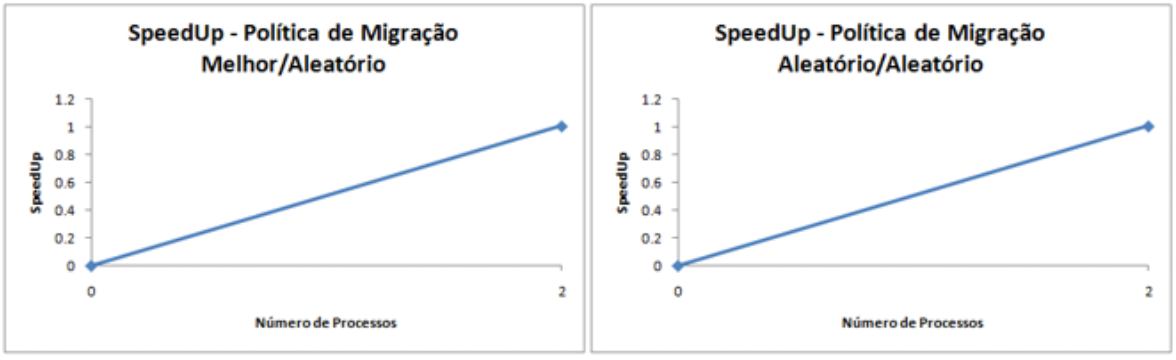

Figura 4.18: Gráficos que apresenta o *SpeedUp* para a estratégia de duas ilhas empregando as políticas de migração *Melhor/Aleatório* e *Aleatório/Aleatório* para a Função 4.

É notório na Figura 4.19 que a eficiência alcançada pelos processos na estratégia de duas ilhas são praticamente iguais, comparando as diferentes políticas de migração. Este resultado é coerente com os resultados obtidos no cálculo do *SpeedUp*, pois como foi possível verificar, o *SpeedUp* das estratégias paralelas para as duas políticas de migração são visivelmente iguais.

Através da Figura 4.20, é possível certificar que as estratégias paralelas possuem melhores soluções média que a solução média encontrada pela abordagem sequencial.

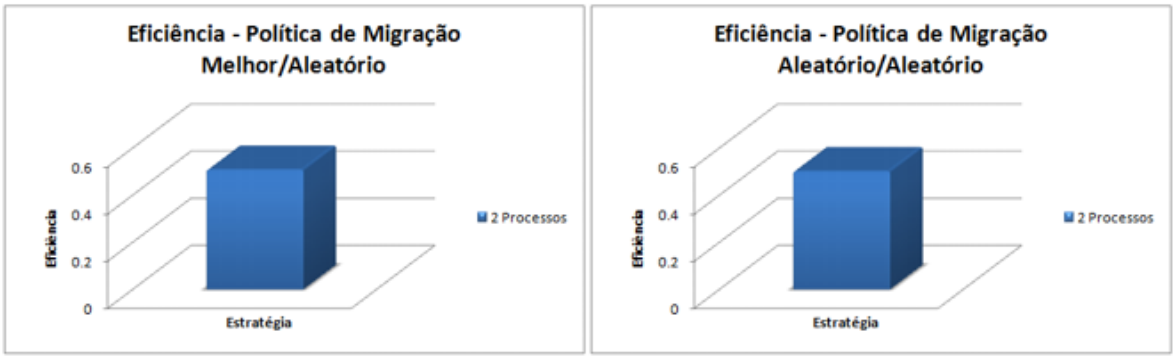

Figura 4.19: Gráficos que apresenta a eficiência para a estratégia de duas ilhas empregando as políticas de migração *Melhor/Aleatório* e *Aleatório/Aleatório* para a Função 4.

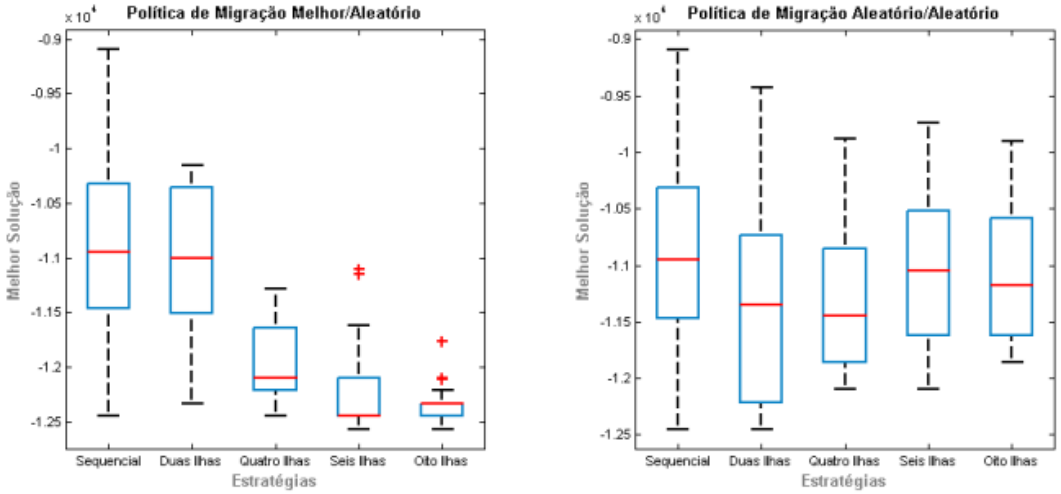

Figura 4.20: Gráficos que apresentam as melhores soluções encontradas para as variações do método, onde o critério de parada é atingir 5000 (cinco mil) gerações, para a Função 4.

Escolhemos as configurações paralelas que obtiveram o melhor valor médio das soluções encontradas para cada política de migração e comparamos sua curva de convergência com a da abordagem sequencial. Podemos verificar que a abordagem sequencial e o modelo com 2 (duas) ilhas, empregando a política de migração *aleatório/aleatório*, convergem mais rápido para uma solução próxima ao mínimo global, mas com isso elas não detêm a melhor solução para o problema, comparados ao modelo com 8 (oito) ilhas, vide Figura 4.21.

Todavia, mesmo que as estratégias paralelas avaliadas não tenham obtido resultados animadores em relação ao número de gerações gastos para a convergência, do tempo computacional gasto e consequentemente do *SpeedUp* e eficiência, todas elas encontraram melhores soluções para o problema teste em questão.

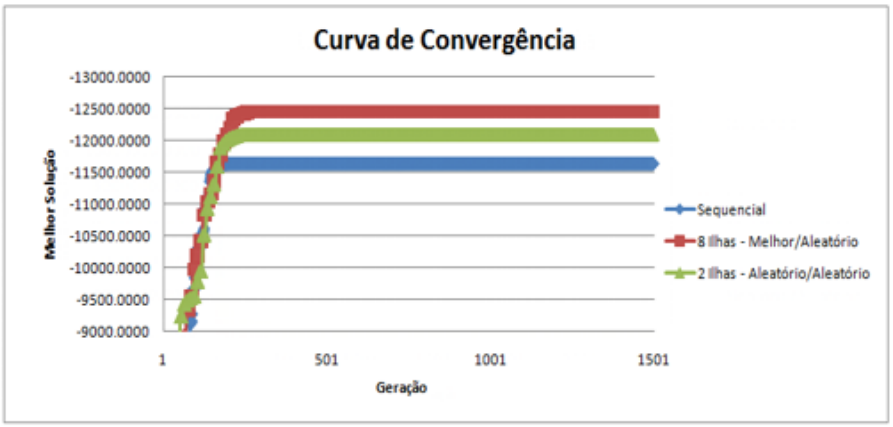

Figura 4.21: Gráficos que apresentam as curvas de convergência das abordagem paralelas e sequencial, para a Função 4.

4.2.5 Função 5

Na Figura 4.22, é exposto que na maioria das execuções dos experimentos, os métodos propostos não convergiram antes que o número máximo de gerações pré-determinado fosse atingido, exceto algumas poucas vezes que o método encontrou uma boa solução. Como podem ser analisadas, estas exceções ocorreram quando foram empregadas seis e oito ilhas utilizando a política de migração *melhor/aleatório*, sendo que, convergiram gastando um pouco mais de 10% das gerações das 5000 (cinco mil) propostas.

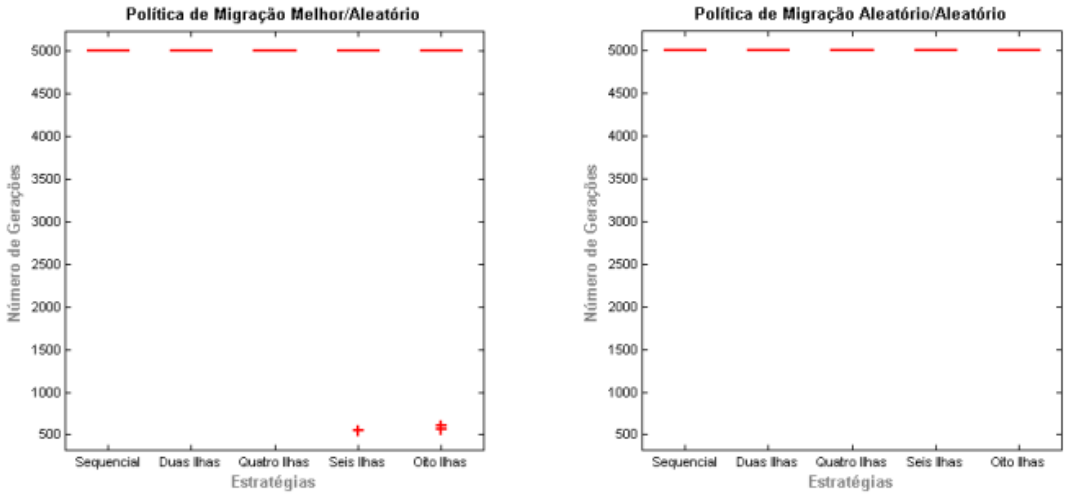

Figura 4.22: Gráficos que apresenta o número médio, mínimo e máximo de gerações gastos em cada método, até que o critério de parada seja atingido, para a Função 5.

Os tempos de execução do método para as configurações que aplicaram a política de

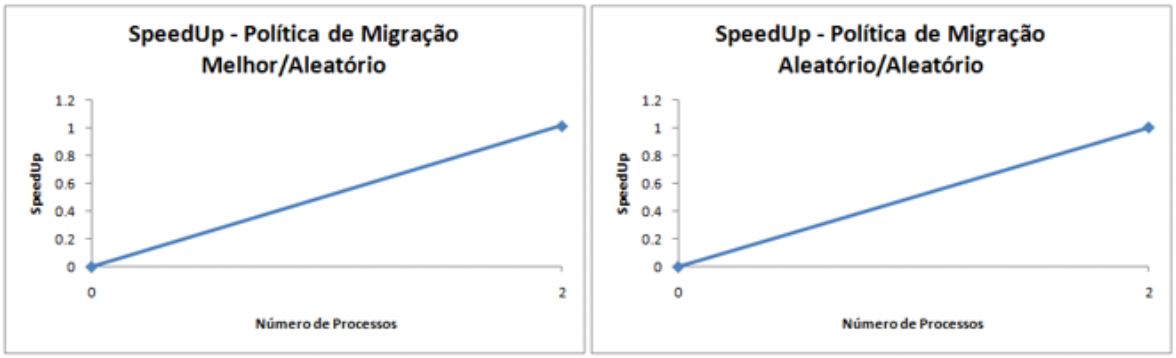

Figura 4.23: Gráficos que apresenta o *SpeedUp* para a estratégia de duas ilhas empregando as políticas de migração *Melhor/Aleatório* e *Aleatório/Aleatório* para a Função 5.

migração *melhor/aleatório*, foram todos melhores que a sua implementação sequencial, vide Tabela 4.17.

	Sequencial	**Duas Ilhas**	**Quatro Ilhas**	**Seis Ilhas**	**Oito Ilhas**
Tempo(ms)	178.059	175.079	166.608	157.514	151.461

Tabela 4.17: Tempo de Execução para a Função 5, até que o critério de parada seja atingido - Política de Migração *Melhor/Aleatório*.

No entanto, nas configurações que utilizaram a outra política, apenas o modelo de duas ilhas teve seu tempo reduzido, como apresenta a Tabela 4.18.

	Sequencial	**Duas Ilhas**	**Quatro Ilhas**	**Seis Ilhas**	**Oito Ilhas**
Tempo(ms)	178.059	177.356	180.165	181.584	184.439

Tabela 4.18: Tempo de Execução para a Função 5, até que o critério de parada seja atingido - Política de Migração *Aleatório/Aleatório*.

Analisando a Figura 4.23, percebemos que o *SpeedUp* para o modelo de duas ilhas empregando as duas políticas de migração são bons resultados e muito próximos entre si.

A Figura 4.24 apresenta a eficiência do modelo de duas ilhas utilizando as políticas de migração propostas.

A Figura 4.25, apresenta gráficos das melhores soluções encontradas por cada configuração do método. A partir dela, observamos que em todos os modelos de ilhas e suas políticas de migração, os métodos possuem médias menores que a abordagem sequencial.

Observamos na Figura 4.26, que as 3 (três) curvas de convergência analisadas, que representam a abordagem sequencial, o modelo com 8 (oito) ilhas empregando a política de migração *melhor/aleatório* e o modelo com 6 (seis) ilhas usando a outra política proposta, se

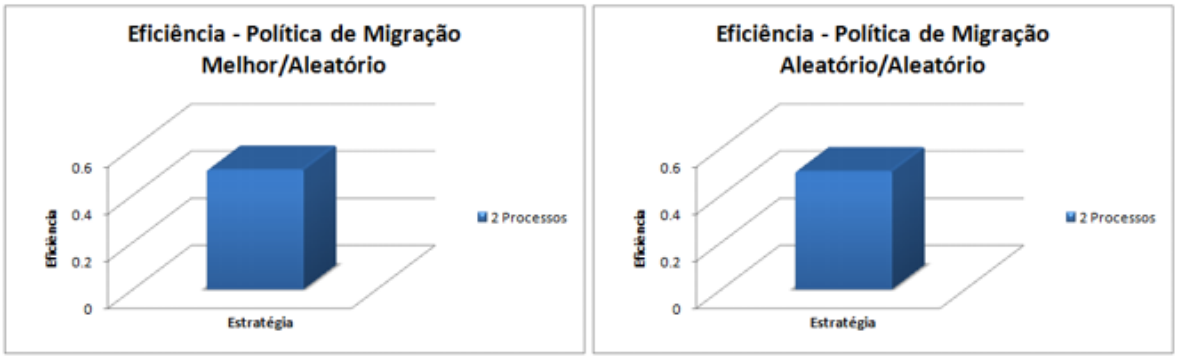

Figura 4.24: Gráficos que apresenta a eficiência para a estratégia de duas ilhas empregando as políticas de migração *Melhor/Aleatório* e *Aleatório/Aleatório* para a Função 5.

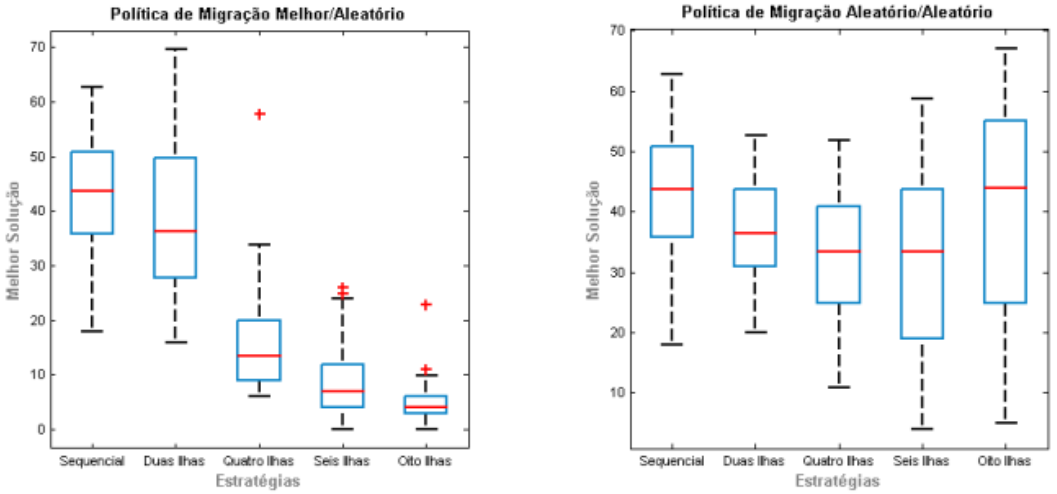

Figura 4.25: Gráficos que apresentam as melhores soluções encontradas para as variações do método, onde o critério de parada é atingir 5000 (cinco mil) gerações, para a Função 5.

assemelham muito. Entretanto, o modelo com 8 (oito) ilhas convergem para um resultado melhor.

Além disso, os métodos convergem muito rapidamente, fazendo com que eles convirjam prematuramente e acabem caindo em uma bacia de atração, onde ficam presos até que critério de parada seja atendido.

Portanto, todos os modelos em ilhas para a abordagem que migra as melhores soluções das supopulações alcançaram bons resultados comparados à versão sequencial do método, com destaque para o modelo de oito ilhas.

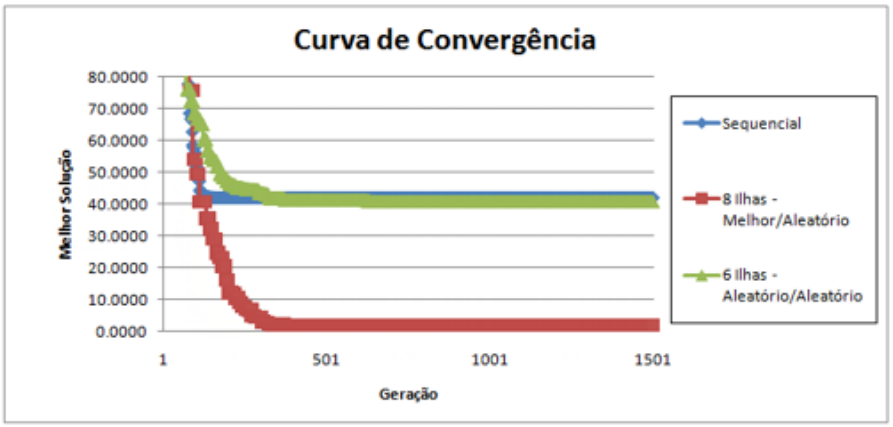

Figura 4.26: Gráficos que apresentam as curvas de convergência das abordagem paralelas e sequencial, para a Função 5.

4.2.6 Função 6

Diferentemente do que vem ocorrendo até o presente momento, as configurações de abordagens paralelas que empregaram a política de migração *aleatório/aleatório*, onde indivíduos escolhidos aleatoriamente na população destino são substituídos por outros indivíduos também com escolha aleatória, foram as que encontraram as melhores soluções mais rapidamente, como apresenta a Figura 4.27.

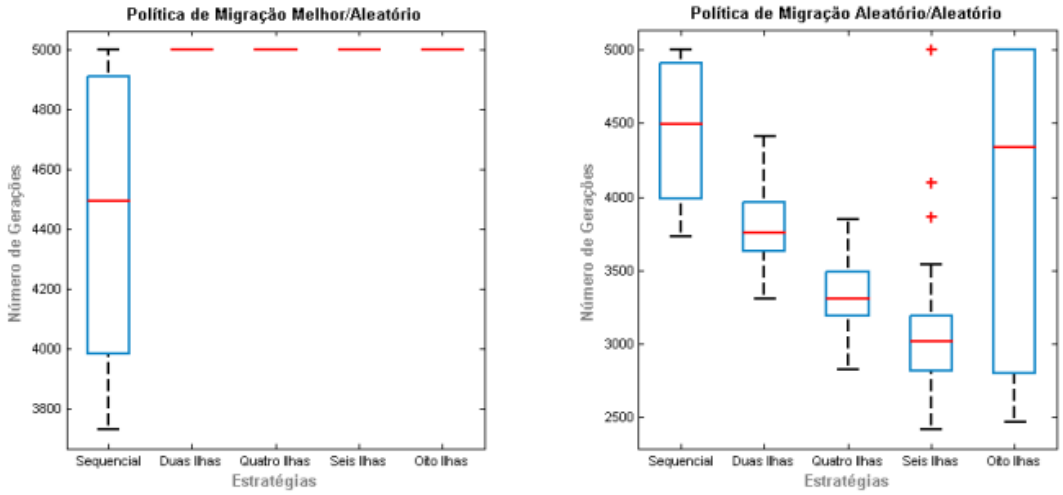

Figura 4.27: Gráficos que apresenta o número médio, mínimo e máximo de gerações gastos em cada método, até que o critério de parada seja atingido, para a Função 6.

As abordagens que empregaram a política *melhor/aleatório* não conseguiram diminuir o tempo de execução com relação a abordagem sequencial, entretanto, as abordagens que usaram

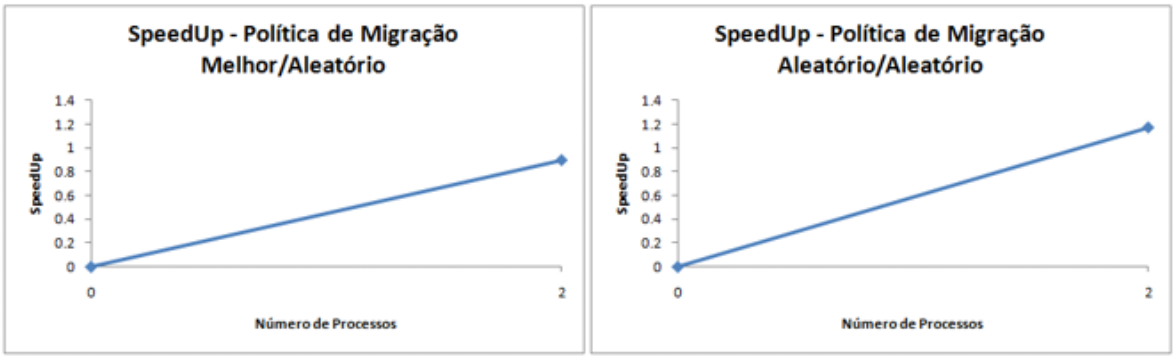

Figura 4.28: Gráficos que apresenta o *SpeedUp* para a estratégia de duas ilhas empregando as políticas de migração *Melhor/Aleatório* e *Aleatório/Aleatório* para a Função 6.

a política *aleatório/aleatório* diminuíram o seu tempo, como represetado nas Tabelas 4.19 e 4.20.

	Sequencial	**Duas Ilhas**	**Quatro Ilhas**	**Seis Ilhas**	**Oito Ilhas**
Tempo(ms)	186.701	208.401	207.761	207.908	208.183

Tabela 4.19: Tempo de Execução para a Função 6, até que o critério de parada seja atingido - Política de Migração *Melhor/Aleatório*.

	Sequencial	**Duas Ilhas**	**Quatro Ilhas**	**Seis Ilhas**	**Oito Ilhas**
Tempo(ms)	186.701	159.011	136.157	130.899	161.476

Tabela 4.20: Tempo de Execução para a Função 6, até que o critério de parada seja atingido - Política de Migração *Aleatório/Aleatório*.

Desta maneira, analisando os gráficos de *SpeedUp* na Figura 4.28, nota-se que o modelo de duas ilhas para a política *aleatório/aleatório* possue a melhor medida.

A melhor medida de eficiência foi resultada da abordagens paralela, que migram soluções aleatórias, como apresentado na Figura 4.29.

Analisando a Figura 4.30, concluímos que as configurações paralelas empregando a política de migração *aleatório/aleatório* geralmente encontram soluções melhores que a abordagem sequencial.

Mas observando de forma detalhada os resultados das abordagens paralelas para a política de migração aleatória, percebemos que o modelo de quatros ilhas encontra a melhor solução para o problema entre todas configurações e a média das soluções encontradas também é a melhor comparada com outras, como apresenta a Tabela 4.21.

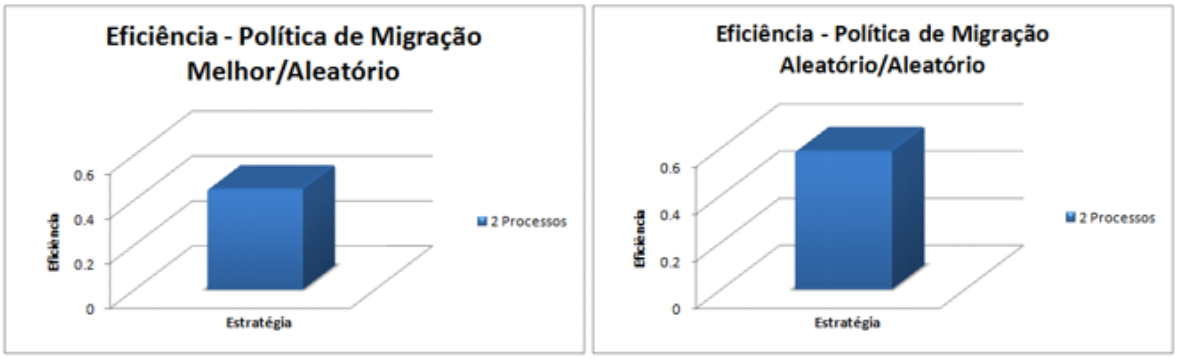

Figura 4.29: Gráficos que apresenta a eficiência para a estratégia de duas ilhas empregando as políticas de migração *Melhor/Aleatório* e *Aleatório/Aleatório* para a Função 6.

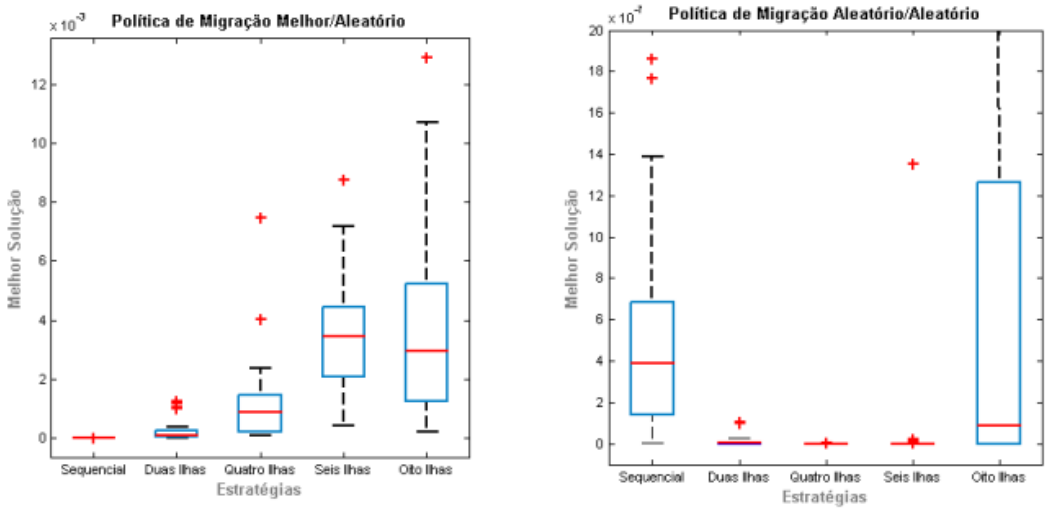

Figura 4.30: Gráficos que apresentam as melhores soluções encontradas para as variações do método, onde o critério de parada é atingir 5000 (cinco mil) gerações, para a Função 6.

Avaliando as configurações paralelas de cada política de migração, que possuem a melhor solução, observamos que o modelo em ilhas que emprega a política de migração *melhor/aleatório* converge prematuramente e acaba estacionando em uma bacia de atração, fazendo como que o método não encontre solução mais próxima ao mínimo, como apresentado na Figura 4.31.

A abordagem sequencial e a outra configuração paralela demoram algumas gerações a mais, comparada com a configuração mencionada acima. No entanto, elas convergem para uma boa solução.

Contudo, o modelo que divide a população total em quatro ilhas possui bons resultados como menor número de gerações, tempo de execução menor que a abordagem sequencial, até que uma boa solução seja encontrada, e melhor solução encontrada para a função de teste.

	Sequencial	Duas Ilhas	Quatro Ilhas	Seis Ilhas	Oito Ilhas
Melhor	$1.42e^{-09}$	$6.63e^{-06}$	$7.37e^{-05}$	$4.06e^{-04}$	$2.14e^{-04}$
Média	$5.42e^{-07}$	$2.50e^{-04}$	$1.20e^{-03}$	0.0036	0.0035
Desvio Padrão	$5.84e^{-07}$	$3.57e^{-04}$	$1.50e^{-03}$	0.0021	0.003

Tabela 4.21: Melhor solução encontrada, média e desvio padrão encontrados, onde o critério de parada é atingir 5000 (três mil) gerações, para a Função 6 - Política de Migração *Melhor/Aleatório*.

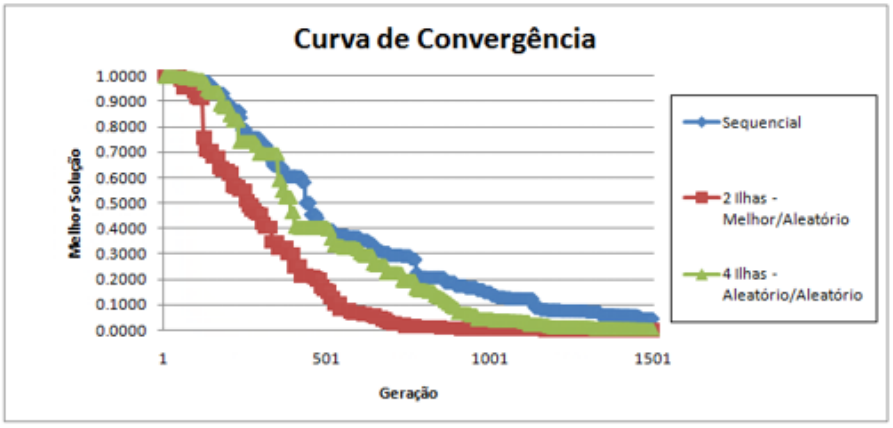

Figura 4.31: Gráficos que apresentam as curvas de convergência das abordagem paralelas e sequencial, para a Função 6.

Capítulo 5

Conclusão

Neste trabalho, desenvolvemos uma abordagem paralela empregando o modelo em ilhas para o algoritmo de Evolução Diferencial Autoadaptativo. A implementação deste modelo foi proposta com o intuito de obter melhor desempenho do método, onde o número de gerações, as soluções encontradas, o tempo e outras medidas de desempenho são analisados e comparados a abordagem sequencial do mesmo.

Para avaliar a solução proposta, foram executados vários experimentos utilizando uma variedade de problemas testes com características distintas, analisando assim, o desempenho do método em diversos cenários.

Os resultados experimentais indicam que esta abordagem paralela proposta cumpre seu objetivo maior, onde busca uma diminuição do número de gerações, o que consequentemente leva a um decréscimo do tempo de execução do algoritmo e uma melhora da solução do problema otimizado. Foi verificado também que, mesmo quando a abordagem proposta não atende seu maior objetivo, o método encontra boas soluções em relação à abordagem sequencial do método.

No entanto, a política de migração empregada e o número de ilhas são fatores que possuem grande influência na convergência do método paralelo. Para os problemas testes avaliados a política de migração, que substitui um bom migrante por um indivíduo aleatório, foi a que levou o método, na maioria dos experimentos, a convergir mais rapidamente e encontrar melhores soluções. Entretanto, este fato não indica que esta política de migração será sempre a melhor solução para a troca de informação entre as subpopulações.

Analisando as medidas de desempenho computacional estudadas, podemos concluir que o modelo com duas ilhas, ou seja, duas subpopulações, apresentaram bons resultados e, desta forma, além de encontrar boas soluções e possuir bom desempenho o método é bem avaliado computacionalmente pelas medidas do *SpeedUp* e eficiência dos processos.

Contudo, ao empregarmos as configurações paralelas corretas a cada problema de otimização, satisfazemos seu maior propósito, que é melhorar a eficiência e desempenho do algoritmo de Evolução Diferencial Autoadaptativo em comparação com sua abordagem sequencial, além

de torná-lo mais simples que o algoritmo de Evolução Diferencial clássico.

Capítulo 6

Referências Bibliográficas

Alba E. and Tomassini M. (2002). Parallelism and Evolutionary Algorithms, IEEE Transactions on Evolutionary Computation, Vol. 6, N^o 5.

Abbass, H. A. (2002). An evolutionary artificial neural networks approach for breast cancer diagnosis, Artificial Intelligence in Medicine 25: 265-281.

Adamidis P. (1998). Parallel Evolutionary Algorithms: A Review, 4th Hellenic-European Conference on Computer Mathematics and its Applications.

Brest, J. and Maucec, M. S. (2006). Control parameters in self-adaptive differential evolution.

Cantú-Paz, E. (2001). Migration Policies, Selection Pressure, and Parallel Evolutionary Algorithms, Journal of Heuristics.

Castro, L. N. (2002). Artificial Immune Systems: A New Computational Intelligence Approach. Springer-Verlag, London.

Chang, C. S., Xu, D. Y. and Quek, H. B. (1999). Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mass transit system, IEE Proceedings on Electric Power Applications 146: 577-583.

Chakraborty U. K. (2008). Advances in Differential Evolution. Springer Publishing Company, Incorporated.

Grama, A., Gupta A., Karypis G., Kumar, V. (2003). Introduction to Parallel Computing, Second Edition, Addison Wesley.

Goldberg D.(1989). Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley.

Goldberg, D.E. and K., Deb. (1991). A Comparative Analysis of Selection Schemes used in Genetic Algorithms. Foundations of Genetic Algorithms. San Mateo, CA: Morgan Kaufmann 1, 6993.

Gropp, W., Kennedy, K., Torczon, L., White, A. (2003). Sourcebook of Parallel Computing, Morgan Kaufmann Publishers.

Ilonen, J., Kamarainen, J.-K. and Lampinen, J. A. (2003). Differential evolution training algorithm for feed-forward neural networks, Neural Processing Letters 17: 93-105.

Liu J. and Lampinen J. (2005). A fuzzy adaptive differential evolution algorithm. Soft Comput.,9:448-462.

Madavan N. K. (2002). Multiobjective optimization using a pareto differential evolution approach. In Proceedings of the Evolutionary Computation on 2002. Proceedings of the 2002 Congress, Washington, DC, USA. IEEE Computer Society.

Magoulas, G. D., Plagianakos, V. P. and Vrahatis, M. N. (2001). Hybrid methods using evolutionary algorithms for on-line training, International Joint Conference on Neural Networks (IJCNN), Vol. 3, Washington, DC, USA, pp. 2218-2223.

Masters, T. and Land, W. (1997). A new training algorithm for the general regression neural network, IEEE International Conference on Systems, Man, and Cybernetics, Vol. 3, IEEE Press, pp. 1990-1994.

Mezura-Montes, Efr n., Velazquez-Reyes J., and Coello Coello C. A. (2006). A comparative study of differential evolution variants for global optimization. In GECCO: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pages 485-492.

Milani A. and Santucci V. (2010). Asynchronous Differential Evolution. IEEE Congress on Evolutionary Computation 2010: 1-7.

Moraes E. F., Alves J. M. C. B., Souza M. J. F., Cabral I. E., Martins A. X. (2006). Um modelo de programação matemática para otimizar a composição de lotes de minério de ferro da mina Cauê da CVRD. Revista da Escola de Minas (Impresso), v. 59, p. 299-306.

Pacheco M. A. C., (1999), Algoritmos Genéticos: Princípios e Aplicações, ICA: Laboratório de Inteligência Computacional Aplicada.

Poli R., Kennedy J., and Blackwell, T. (2007). Particle swarm optimization. Swarm Intelligence,33-57.

Qin, A. K., Huang, V. L., and Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. Trans. Evol. Comp, 13(2):398417.

Qin, A. K. and Suganthan, P. N. (2005). Self-adaptive differential evolution algorithm for numerical optimization. In IEEE Congress on Evolutionary Computation (CEC 2005), pages 17851791. IEEE Press.

Reddy, M. J. and Kumar, D. N. (2007). Multiobjective differential evolution with application to reservoir system optimization, Journal of Computing in Civil Engineering 21: 136-146.

Schwefel, H.P. (1995). Evolution and Optimum Seeking. Wiley, New York.

Silva G. P., Souza M. J. F., Reis J. V. A. (2007). Análise comparativa de métodos para resolver o problema de programação de tripulações. Revista Produção Online, v. 7, p. 126-143.

Souza M. J. F., Mine M. T., Silva M. S. A., Silva G. P. (2006). Iterated Local Search Aplicado à Resolução do Problema de Programação de Jogos do Campeonato Brasileiro de Futebol. Revista da Pesquisa e Pós-Graduação, v. 6, p. 48-52.

Storn R. (1999) System design by constraint adaptation and differential evolution,. IEEE Transactions on Evolutionary Computation, 3:22-34.

Storn, R. M. (1999). Designing digital filters with differential evolution, in D. Corne, M. Dorigo and F. Glover (eds), New Ideas in Optimization, Advanced Topics in Computer

Science, McGraw-Hill Inc., pp. 109-125.

Storn R. and Price K. (1995). Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces, Journal of Global Optimization.

Storn R. and Price K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11:341-359.

Tasoulis D. K., Pavlidis N., Plagianakos V. P., and Vrahatis, M. N. (2004). Parallel differential evolution. In In IEEE Congress on Evolutionary Computation (CEC).

Teo, J. (2006). Exploring dynamic self-adaptive populations in differential evolution. Soft Comput., 10(8):673686.

Tomassini M. (1999). Parallel and distributed evolutionary algorithms: A review.

World Community Grid (2010). www.worldcommunitygrid.org, site acessado em 20 de Setembro de 2010.