Rodrigo da Cruz Alvarenga Fajardo Pontes

Introdução de um índice de desempenho para a migração no modelo de ilhas

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional, da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre em Modelagem Computacional.

Orientador: Prof. D.Sc. Leonardo Goliatt da Fonseca

Juiz de Fora

2020

Rodrigo da Cruz Alvarenga Fajardo Pontes,

Introdução de um índice de desempenho para a migração no modelo de ilhas/ Rodrigo da Cruz Alvarenga Fajardo Pontes. – Juiz de Fora: UFJF/MMC, 2020.

VIII, 52 p.: il.; 29, 7cm.

Orientador: Leonardo Goliatt da Fonseca

Dissertação (mestrado) – UFJF/MMC/Programa de Modelagem Computacional, 2020.

Referências Bibliográficas: p. 50 – 52.

1. modelo de ilhas. 2. índice de desempenho. 3. computação evolucionista. I. , Leonardo Goliatt da Fonseca. II. Universidade Federal de Juiz de Fora, MMC, Programa de Modelagem Computacional.

Rodrigo da Cruz Alvarenga Fajardo Pontes

Introdução de um índice de desempenho para a migração no modelo de ilhas

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional, da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre em Modelagem Computacional.

Aprovada em 2 de Março de 2020.

BANCA EXAMINADORA

Prof. D.Sc. Leonardo Goliatt da Fonseca - Orientador
Universidade Federal de Juiz de Fora

Prof. D.Sc Carlos Cristiano Hasenclever Borges
Universidade Federal de Juiz de Fora

Prof. D.Sc. Eduardo Krempser da Silva
Fiocruz

AGRADECIMENTOS

Eu agradeço primeiramente à minha noiva Jéssica por estar me apoiando durante toda a realização deste trabalho. Agradeço também ao programa, por todo o apoio que me foi concedido nesses anos de trabalho. Agradeço também à minha família, que está sempre ao meu lado em todos os momentos da minha vida.

RESUMO

Nesse trabalho é acrescentado ao modelo de ilhas um índice de desempenho para avaliar o quão eficiente a ilha está sendo em resolver problemas de otimização sem restrições. As ilhas com índices maiores recebem mais indivíduos quando a migração ocorre. Na ocasião da aplicação do operador de migração, são selecionados alguns indivíduos para migrarem de uma determinada ilha. Esses indivíduos escolhem então suas ilhas de destino, ou se permanecem na ilha em que estão, com mais chances de escolher uma ilha que possui um alto índice de desempenho. As simulações realizadas indicam que o modelo proposto apresenta resultados semelhantes aos resultados gerados pelo melhor algoritmo indicado pela literatura para cada problema. Percebeu-se também que ao retirar o algoritmo mais eficiente do modelo, o modelo proposto consegue se adaptar e gerar soluções eficientes, utilizando-se das características dos algoritmos restantes.

Palavras-chave: modelo de ilhas. índice de desempenho. computação evolucionista.

ABSTRACT

In this work, a performance index was added to the island model, to evaluate how efficiently the population of an island is at solving a given problem. The islands with higher indexes receive more individuals when migration occurs. when the migration operator is used, a few individuals are chosen to make a decision: either go to another island or stay in the current island. The individual has a higher chance of choosing an island with a high performance index. After running the simulations, we noticed that the solutions of the new model were as good as the solutions from the best algorithm for each problem. We also noticed that even if we remove the most efficient algorithm from the model, it manages to adapt and still provide efficient solutions, making use of the characteristics from the remaining algorithms.

Keywords: island model. performance index. evolutionary computation.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

1 INTRODUÇÃO

Computação Evolucionista [1] é uma área da inteligência artificial, que apresenta maneiras de encontrar soluções para problemas, que geralmente são formulados como problemas de otimização. É uma família de diferentes algoritmos, inspirados na teoria da seleção natural. Esses algoritmos, denominados algoritmos evolucionistas, são caracterizados por implementar os mecanismos da evolução biológica, como reprodução, mutação, recombinação e seleção.

Os primeiros registros da utilização de métodos de computação evolucionista para a resolução de problemas começaram aproximadamente na década de 1950. Naquela época, já haviam evidências da existência de modelos que nos ajudavam a entender o processo natural da evolução. Algumas das primeiras descrições de um processo evolutivo apareceram nos artigos escritos por Friedberg em 1958 [2]. Isso representou um dos primeiros trabalhos em aprendizado de máquina e no uso de um algoritmo evolutivo. A estrutura básica de um algoritmo evolucionista é mostrada na Figura 1.1.

Estrutura do algoritmo
Início
t=0
Inicializar os indivíduos P(t)
Aplicar a função de avaliação na população P(t)
Enquanto critério de parada não é alcançado **faça**
Selecionar a nova população
Aplicar operadores evolutivos
Fim enquanto
Fim

Figura 1.1: Estrutura dos algoritmos evolucionistas

No mesmo período, Bremermann também aplicou a evolução para resolver problemas de otimização [3]. Ele desenvolveu algumas das primeiras teorias de algoritmos evolutivos, mostrando que a taxa de mutação ótima para problemas linearmente separáveis era de $1/l$, no caso de indivíduos com l bits de codificação.

Esses e outros esforços não foram reconhecidos à época, devido ao fato de ser uma área nova que estava começando a aparecer no meio acadêmico. Porém, na metade da

década de 1960, o estudo começou a se popularizar mais no meio acadêmico e começaram a surgir as raízes do que conhecemos hoje como algoritmos evolutivos. Dentre algumas formas de algoritmos evolutivos que surgiram na época, a programação evolucionista foi desenvolvida por Fogel [4], e as bases dos algoritmos genéticos foram desenvolvidas por Holland [1].

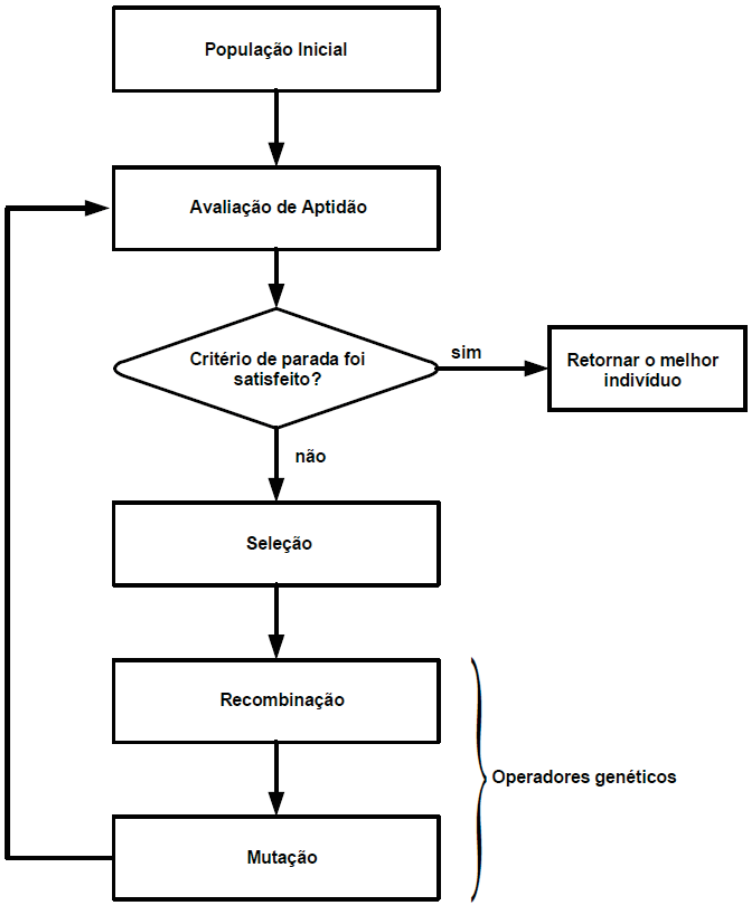
Nos anos 60, foi desenvolvida a programação evolutiva, os algoritmos genéticos e as estratégias de evolução [5]. A programação evolutiva foi desenvolvida por Lawrence J Fogel em 1960. Naquele período, a inteligência artificial estava restrita em heurísticas e nas simulações de redes neurais primitivas. Fogel acreditava que essas abordagens eram limitadas pois elas buscavam imitar os seres humanos como modelos, ao invés de focar no processo que cria os seres inteligentes, a evolução. Fogel acreditava que a inteligência consistia na adaptação do comportamento dos seres com a finalidade de atingir objetivos e superar barreiras encontradas no meio em que vivem. Sua proposta foi a seguinte: Uma população é exposta ao meio, e avaliada para saber o valor da "fitness"de cada indivíduo. Os melhores indivíduos geram filhos que seriam gerados através de uma mutação aleatória realizada nos pais. Esse processo é repetido até que um critério de parada seja atingido.

As primeiras ideias sobre algoritmos genéticos estão presentes nos trabalhos de Holland na década de 60. Essas ideias eram baseadas em sistemas que teriam a habilidade de se modificar para se adaptar melhor ao meio em que estão inseridos. Para Holland, a principal vantagem desses modelos é o uso da competição e inovação, que dão à população a habilidade de responder a mudanças inesperadas no meio.

Já na década de 70, já existia um interesse entre os pesquisadores de entender como a variação dos parâmetros afetaria o funcionamento e a qualidade das soluções dos algoritmos genéticos. De Jong, em 1975, analisou os efeitos da variação dos vários parâmetros de um algoritmo genético (tamanho da população, crossover, mutação, etc) [6].

Os algoritmos evolutivos mais populares são os algoritmos genéticos [7], descritos no artigo *A genetic algorithm tutorial* [8], que são baseados em uma população formada por possíveis soluções (cromossomos que representam as soluções possíveis de um determinado

problema no espaço dos genótipos). Os indivíduos da população são inicializados, geralmente aleatoriamente, e então passam por um ciclo evolutivo até que um critério de parada seja satisfeito, que é geralmente encontrar uma solução próxima o suficiente a um valor ideal, ou então um número máximo de gerações ou de avaliações da função objetivo. Esse ciclo envolve a determinação da aptidão (função de mérito) de todos os indivíduos da população, que é geralmente o valor da função objetivo do problema de otimização que está sendo resolvido, a seleção de um grupo de invidíduos para dar origem a uma nova geração, e também a aplicação de alguns operadores genéticos (ou operadores evolutivos), como recombinação e mutação. A recombinação é seleção de quais partes do material genético dos dois pais irão dar origem ao filho, que será uma combinação dos pais escolhidos. A mutação é a alteração de partes do genótipo de alguns indivíduos para que possíveis melhorias possam ocorrer, embora a maioria delas não contribua e seja descartada. Na Figura 1.2, podemos observar como é a estrutura de um algoritmo genético tradicional.



Fonte: autor

Figura 1.2: Estrutura de um algoritmo genético

Um grande número de algoritmos inspirados pela natureza foram introduzidos nas últimas décadas, muitos usando sistemas distribuídos para diminuir o tempo de simulação [9]. Os modelos distribuídos foram um grande avanço na área, pois com eles foi possível reduzir consideravelmente o tempo de simulação, proporcionalmente ao número de processadores, e portanto abriu-se a possibilidade de atacar problemas mais complexos. Eles funcionam através da divisão da população em subpopulações, cada uma sendo simulada em um processador, que evoluem separadamente. A partir daí, surgiram alguns desafios a serem estudados em cima dos modelos distribuídos, como por exemplo, como será feita a comunicação entre os processadores através de uma rede de memória compartilhada.

Um modelo distribuído que ficou bastante conhecido foi o modelo mestre-escravo [10], no qual um processador, denominado mestre, é responsável por dividir e destribuir a população entre os outros processadores, denominados escravos. Após estes avaliarem suas subpopulações e aplicarem os operadores, eles devolvem os resultados ao mestre para que o ciclo continue.

Caso o modelo mestre-escravo utilize uma máquina com memória distribuída, a comunicação entre os processos precisa ser bem eficiente, porque senão o método pode perder eficiência. Um grande problema desse algoritmo é o fato de ter um gargalo na comunicação entre os processadores, pois o mestre fica com um tráfego muito grande de informações caso haja um número grande de processadores. Então o modelo pode ficar limitado em relação à escalabilidade, ou seja, ao aumentar muito o número de processadores, não há uma mudança significativa em desempenho. Uma análise mais profunda sobre este algoritmo, em relação ao número ideal de processadores entre outros aspectos é apresentada em [11]. A seguir é apresentado o pseudocódigo de um algoritmo mestre-escravo:

Algoritmo mestre-escravo

Início

Definir o número de indivíduos e inicializar a população

Dividir a população entre os processadores

Enquanto critério de parada não é alcançado **faça**

Para $i = 1, numeroProcessadores$ **faça**

Mestre envia os indivíduos para o processador de índice i

Fim para

Escravos fazem a avaliação dos indivíduos e aplicam os operadores evolutivos

Para $i = 1, numeroProcessadores$ **faça**

Enviar os resultados do processador i para o mestre

Fim para

Fim enquanto

Fim Fonte: autor

Figura 1.3: Algoritmo mestre-escravo

Um dos desafios encontrados ao tentar resolver problemas utilizando-se de algoritmos genéticos, é que em alguns casos, o algoritmo percorre um caminho específico no espaço de soluções até encontrar um máximo ou mínimo local, o que leva a convergência prematura. Isso ocorre porque alguns problemas possuem diversos locais de máximo ou mínimo que são distantes da solução ótima. Então, quando indivíduos começam a se distanciar do ótimo local, eles são considerados piores e são descartados, mesmo se estiverem indo em direção à solução ótima.

Na Figura 1.4, podemos ver uma função que possui o mínimo global, e também um mínimo local. Perceba que nesta situação há a possibilidade da solução não chegar na solução ótima, pois como o mínimo local é o melhor ponto em sua redondeza, é possível que a solução fique estagnada neste ponto.

Fonte: autor

Figura 1.4: Função que apresenta um mínimo local

Para resolver esse problema, foram criados alguns diferentes modelos, dentre eles o modelo distribuído de ilhas, que é amplamente usado em diversas áreas. Uma análise profunda deste modelo encontra-se em [12]. No modelo de ilhas, a população é dividida entre várias ilhas que evoluem separadamente, geralmente com um algoritmo diferente para cada ilha, produzindo diferentes soluções baseadas nas características de seu algoritmo. Além disso, a existência de várias subpopulações ajuda a preservar a diversidade genética, pois cada ilha pode seguir uma trajetória diferente no espaço de busca. Uma outra vantagem deste modelo é a facilidade de paralelização, pois pode-se atribuir cada ilha a um processador, e assim diminuir significativamente o tempo de simulação. O proposta apresenta em [13] estuda um modelo que é uma variação do modelo de ilhas. Já em [14], o modelo de ilhas é tolerante a falhas, portanto pode ser simulado em uma estrutura distribuída não confiável.

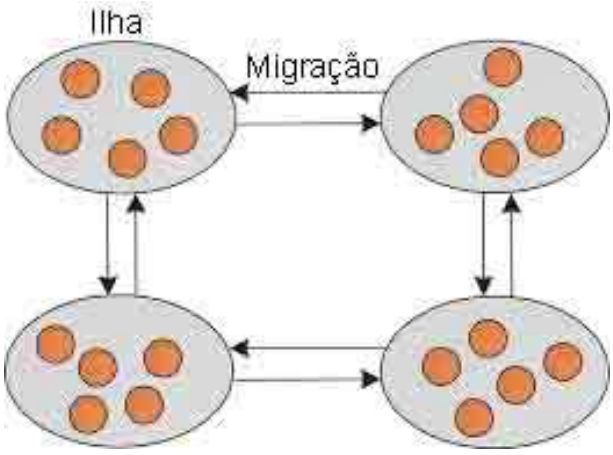
Com a utilização deste modelo, foi possível encontrar soluções melhores para problemas onde havia uma estagnação em um máximo ou mínimo local. Existem vários algoritmos genéticos diferentes que podem ser escolhidos na literatura para serem utilizados em seu modelo de ilhas [15]. Como exemplo podemos citar: Simple Genetic Algorithm (SGA) [7], Differential Evolution (DE) [16], Particle Swarm Optimization (PSO) [17] , Evolution Strategies (ES) [18] e Harmony Search (HS) [19]. Cada um deles foca em um aspecto diferente, como diversidade e qualidade das soluções. É difícil haver um algoritmo que nos dá tanto boas soluções como também uma grande diversidade de material genético, por isso a combinação deles no modelo de ilhas nos dá uma solução muito boa, pois são utilizadas todas as difentes qualidades de cada um dos algoritmos. Existem várias variações do modelo de ilhas, todas buscando melhorias nas soluções finais em relação ao

modelo tradicional. Um exemplo é o modelo estudado em [20], onde os indivíduos que são diferentes do resto da população são alocados para outros processadores, para que não sejam atrapalhados pelos outros indivíduos da população.

De tempos em tempos, migrações irão ocorrer, quando indivíduos vão de uma ilha para outra com o objetivo de compartilhar material genético com outras populações. Isso ajuda a melhorar a qualidade das soluções geradas pelas ilhas. Os estudos apresentados em [21] mostram que, para alguns problemas, a utilização do operador de migração tem um grande efeito nas soluções finais, principalmente a variação da frequência de migração. Ao implementar o operador de migração, deve-se escolher se este será síncrono ou assíncrono, ou seja, se todas as ilhas irão esperar a avaliação de todas as subpopulações para que possam ir para a próxima geração. No caso do modelo proposto nesse trabalho, será utilizada a migração síncrona.

No modelo de ilhas tradicional, a taxa de migração entre duas ilhas é fixa, ou seja, o número de indivíduos indo de uma ilha para outra é sempre o mesmo durante a totalidade da simulação. O presente trabalho busca, por meio da criação de um índice de desempenho, availar o desempenho de cada uma das ilhas do modelo para determinar quais são as ilhas mais eficientes, e assim modificar o operador de migração para que as melhores ilhas fiquem com mais indivíduos, esperando que isso melhore a qualidade da solução final.

Na Figura 1.5, podemos ver um modelo com 4 ilhas, onde cada ilha possui alguns indivíduos (soluções). Neste caso, os indivíduos migram de uma ilha para a outra em um sentido circular, onde apenas as ilhas vizinhas são visíveis para os indivíduos. Essa é a chamada topologia de anel.]



Fonte: autor

Figura 1.5: Modelo de ilhas - topologia de anel

A topologia de migrações pode ter um grande impacto no resultado final das simulações, como mostrado em [22]. Várias topologias podem ser usadas, as mais comuns são a topologia de anel, e também a topologia do grafo completo, onde todas as ilhas estão conectadas com todas as outras. Existem vários estudos sobre modelos onde a topologia de migrações varia, ou seja, modelos dinâmicos que apresentam resultados muitas vezes melhores do que os modelos tradicionais. Como exemplo de modelos deste tipo, podemos citar o modelo estudado em [23].

O modelo de ilhas é utilizado com vários fins, sendo um dos modelos mais eficientes na área de computação evolucionsta. Um exemplo de aplicação está descrito na referência [24], que utiliza o modelo para lidar com problemas de restauração de imagens. A tabela a seguir mostra artigos publicados sobre o modelo de ilhas, em ordem cronológica.

Ano	Autor	Título	Descrição
1989	H. Mühlenbein	Parallel genetic algorithms, population genetics and combinatorial optimization [ühlenbein]	Criação de um modelo paralelo chamado ASPARAGOS
1996	Yen-Wei Chen	A parallel genetic algorithm based on the island model for image restoration [24]	Criação de um modelo onde a população é dividida em subpopulações para resolver um problema de restauração de imagens
1999	Whitley, D.	The island model genetic algorithm: on separability, population size and convergence [25]	Análise de um modelo de ilhas evolutivo
2000	Piotr Jedrzejowicz	An island based evolutionary algorithm for maximizing schedule reliability [26]	Estudos sobre agendamento de tarefas em múltiplos processadores utilizando-se do modelo de ilhas
2004	Zbigniew Skolicki	Improving evolutionary algorithms with multi-representation island models [27]	Um modelo de ilhas com representações diferentes em cada ilha
2005	Zbigniew Skolicki	The influence of migration sizes and intervals on island models [21]	Análise do operador de migração no modelo de ilhas
2006	Steven Gustafson	The Speciating Island Model: An alternative parallel evolutionary algorithm [20]	Novo modelo de ilhas com alocação de indivíduos para novos processadores
2007	Hidalgo, J. I.	Is the island model fault tolerant? [14]	Estudo sobre a tolerância de falhas do modelo de ilhas

Tabela 1.1: Tabela de trabalhos feitos na área (parte 1)

Ano	Autor	Título	Descrição
2008	Changhe Li	An Island Based Hybrid Evolutionary Algorithm for Optimization [28]	Criação de um novo modelo de ilhas híbrido, baseados nos algoritmos PSO, FEP e EDA.
2010	Frédéric Lardeux	A Dynamic Island-Based Genetic Algorithms Framework [29]	Um modelo de ilhas com variação na probabilidade de migrações de acordo com seu impacto na ilha
2013	Tanabe, R.	Evaluation of a randomized parameter setting strategy for island-model evolutionary algorithms [12]	Estudo de estratégias de definição de parâmetros no modelo de ilhas
2017	Grasiele Regina Duarte	A dynamic migration policy to the Island Model [23]	Estudo de uma política de migração dinâmica no modelo de ilhas
2018	Qinxue Meng	Dynamic island model based on spectral clustering in genetic algorithm [30]	Um novo modelo ilhas dinâmico onde o número de subpopulações varia
2019	Grasiele Regina Duarte	Política de migração para Metaheurísticas híbridas usando Modelo Paralelo de Ilhas [31]	Estudo de políticas de migração para o aperfeiçoamento do modelo de ilhas

Tabela 1.2: Tabela de trabalhos feitos na área (parte 2)

1.1 Objetivos

O objetivo deste trabalho é criar e testar um novo modelo de ilhas, que é caracterizado pela alteração no operador migração, fazendo com que algumas ilhas recebam mais indivíduos do que outras. A escolha dessas ilhas será baseada em um índice de desempenho, que mostra o quão bem a ilha está sendo em resolver o determinado problema. Com essa variação no tamanho das populações das ilhas, espera-se que a solução final seja melhor do que modelos tradicionais.

Para isso é preciso mostrar, através dos resultados das simulações, que este novo modelo que contém um índice de desempenho é melhor do que os modelos que usam somente um único algoritmo. Para a implementação do código das simulações, foi utilizada a ferramenta pygmo, que é a versão para Python da biblioteca pagmo, uma biblioteca científica que possui diversos problemas de otimização e também algoritmos para serem utilizados nas simulações. Esta biblioteca é construída com base no paradigma do modelo de ilhas, o que significa que a execução dos algoritmos é automaticamente paralelizada, sem a necessidade do usuário se preocupar com a questão da paralelização. A biblioteca disponibiliza ao usuário a opção de trabalhar com problemas mono-objetivo ou multi-objetivo, mas no caso deste trabalho, apenas serão utilizados problemas mono-objetivos. A referência [32] explica em mais detalhes o funcionamento da biblioteca.

Após a escolha das ferramentas que serão utilizadas, é necessário pensar em como serão as fórmulas para calcular o índice de desempenho que irá avaliar as ilhas para determinar, a cada geração, quais ilhas estão se saindo melhor em resolver o dado problema, e em seguida implementá-las. Como a população das ilhas irá variar de acordo com o índice de desempenho das mesmas, é necessário que a migração entre as ilhas seja feita manualmente, pois a biblioteca pygmo não nos oferece a possibilidade de modificar os parâmetros para que o número de indivíduos que migram para determinada ilha seja customizado para cada uma das ilhas.

Após a etapa de elaboração de fórmulas, é necessário escolher quantas ilhas serão utilizadas no modelo (podendo ser um número de ilhas variável), e também quais algoritmos evolutivos serão utilizados em cada uma das ilhas. A metologia escolhida foi primeiramente utilizar apenas 1 ilha com 1 algoritmo, variando o algoritmo escolhido.

Dessa forma, teremos as medidas de desempenho para cada um dos algoritmos trabalhando sozinho. Em seguida, vamos aumentando o número de ilhas gradativamente para que possamos medir o desempenho dos cenários com mais ilhas, podendo assim fazer uma comparação dos diferentes cenários.

1.2 Estrutura da Dissertação

O Capítulo 2 descreve como o novo modelo é proposto, com base no modelo de ilhas existente. Nele, são apresentados todos os detalhes sobre como o modelo é implementado, quais são suas características, quais fórmulas são utilizadas para calcular o índice de desempenho que é usado para avaliar as ilhas. Além disso, também são apresentados todos os algoritmos utilizados no trabalho, juntamente com seus parâmetros, e também o conjunto de problemas utilizados para testar o novo modelo. Em resumo, é o capítulo que contém todas as informações sobre a implementação do novo modelo para alguém que queira reproduzir os resultados do trabalho futuramente.

O Capítulo 3 apresenta os resultados obtidos através de cada etapa de simulações, os testes estatísticos feitos para avaliar o quão eficientes são os diferentes cenários criados no trabalho, a comparação dos algoritmos evolutivos com o modelo proposto, para ver se este apresenta resultados satisfatórios em comparação com os outros, os resultados do teste de perfil de desempenho para que o melhor cenário possa ser facilmente identificado. Este capítulo apresenta todas as informações necessárias para que possam ser tiradas as conclusões sobre a eficiência do novo modelo proposto.

Por fim, o Capítulo 4 apresenta as conclusões do trabalho. Os resultados serão analisados e, a partir deles, será possível definir se vale a pena usar o modelo proposto para resolver um determinado problema, ao invés de utilizar algum algoritmo dentre os apresentados. Caso este novo modelo apresente soluções de qualidade igual ou superior às soluções do melhor algoritmo para cada um dos 30 problemas, poderemos concluir que vale a pena usar o novo modelo, ao invés de experimentar os algoritmos um a um até encontrar o mais eficiente para o problema.

2 MÉTODOS

2.1 Modelo de ilhas

Um grande problema dos algoritmos evolutivos foi sempre as situações nas quais o algoritmo alcançava um ótimo local. Para tentar resolver esse problema, foram propostas diversas soluções, dentre elas o modelo de ilhas, no qual a população é sub-dividida entre várias sub-populações que evoluem separadamente, com a aplicação de diversos operadores como a seleção, crossover, mutação, e também o operador de migração, através do qual soluções (indivíduos) migram de uma população à outra para que material genético seja compartilhado, com o objetivo de que mais caminhos sejam percorridos no espaço de busca, aumentando as chances de que o ótimo global seja encontrado.

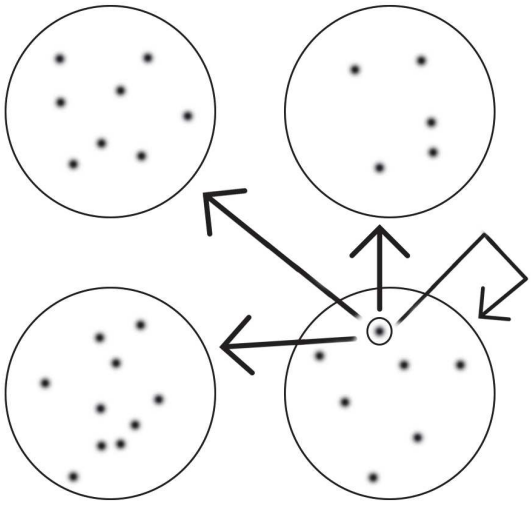
Estudos mostram também que esse novo modelo consegue atingir melhores soluções em menos tempo [25], devido ao fato de que ele pode ser paralelizado facilmente, e de que podem haver diversos caminhos diferentes no espaço de solução entre as sub-populações. Isso ocorre porque no modelo de ilhas tradicional, cada sub população geralmente está sujeita a um algoritmo evolutivo diferente, para que se possa tirar proveito das características diferenciadas de cada um dos algoritmos.

2.2 Abordagem Proposta

Para este trabalho, é proposta uma nova abordagem do modelo de ilhas, na qual o operador de migração é modificado para que haja uma variação na taxa de migração. Isso ocorre com a introdução de um índice de desempenho, que avalia cada uma das ilhas presentes no modelo para determinar o quão eficiente elas estão sendo em resolver o determinado problema. As ilhas que possuírem altos índices de desempenho irão receber mais indivíduos na hora da migração, fazendo com que suas populações aumentem em relação às ilhas que possuem índices menores (ilhas menos eficientes). Espera-se que a consequência disso seja que ao final da simulação, o novo modelo que se utiliza do índice de desempenho apresente bons resultados em comparação com outros modelos.

No momento em que o operador de migração for aplicado, alguns indivíduos da ilha serão selecionados para fazerem uma escolha, onde eles irão decidir para qual ilha irão, ou se continuarão na mesma ilha em que estão. As ilhas que possuem um índice de desempenho alto, serão mais "atraentes"do que as outras, fazendo com que as chances do indivíduo escolher ela sejam maiores. Para evitar que uma determinada ilha fique com um número de indivíduos menor do que o necessário para que seu algoritmo funcione corretamente, foi estabelecido uma quantidade mínima de indivíduos para a população das ilhas, levando em conta o algoritmo evolutivo que está sendo utilizado na ilha.

Ao realizar a migração, os indivíduos que irão fazer a escolha de para qual ilha irão migrar, serão sempre os primeiros indivíduos das ilhas, ou seja, os mais aptos. Isso faz com que, para as ilhas que produzem resultados bons, após a migração ser realizada, a aptidão média da ilha caia um certo valor, pois os melhores indivíduos não estão mais lá. Já as ilhas que não produzem resultados tão bons, apesar de perderem seus melhores indivíduos na migração, irão receber indivíduos melhores ainda provenientes de ilhas melhores. Então, nesse caso não serão prejudicadas.



Fonte: autor

Figura 2.1: Escolha do indivíduo (migração)

2.2.1 Cálculo do Índice de Desempenho

Para calcular o índice de desempenho, é necessário calcular quanto vale o retorno de uma determinada ilha em um determinado instante de tempo. Esse retorno estima qual

será o ganho em termos de qualidade da solução, caso um indivíduo migre para a ilha em questão.

Para o cálculo do retorno, é preciso a cada geração, calcular a média da função objetivo de todos os indivíduos da população da ilha, denominada $\hat{f}$. Esses valores da média serão guardados em uma lista que irá conter valores desde o início da simulação até a geração atual. A razão entre dois valores consecutivos dessa lista $\frac{media[x]}{media[x+1]} = \frac{\hat{f}_i}{\hat{f}_{i+1}}$ é uma medida do quanto a ilha melhorou (ou piorou) de uma geração para a geração seguinte. Portanto, para calcular o índice de desempenho R, foi utilizada a seguinte fórmula:

$$R_n = \sum_{i=0}^{i<n} \log \frac{media[i]}{media[i+1]} = \sum_{i=0}^{i<n} \log \left(\frac{\hat{f}_i}{\hat{f}_{i+1}} \right). \tag{2.1}$$

Para cada geração n, o índice de desempenho R_n é igual ao somatório dos logaritmos da razão entre a média em uma determinada geração e a média na geração seguinte, desde o início da simulação até a presente geração. Quando a migração é realizada, o valor do índice de desempenho da geração anterior à migração para a geração posterior a ela é não é calculado.

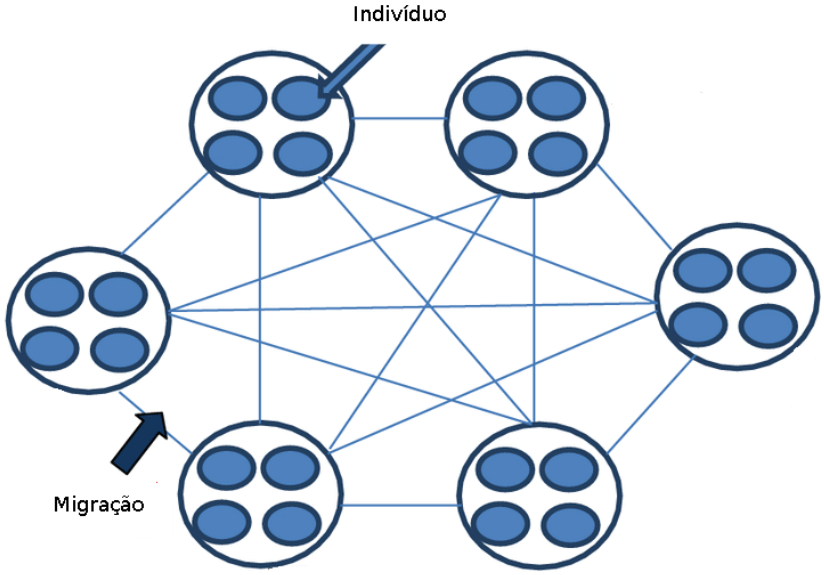
Ao calcular o índice de desempenho, podemos levar em consideração diversos fatores, entre eles o *upside* (subida) e o *downside* (descida). O *upside* equivale às melhorias da média das soluções ao longo das gerações, enquanto que o *downside* equivale às pioras das soluções. A soma dos valores de *upside* ao longo das gerações pode ser considerado como o valor do retorno da ilha para qual é calculado, enquanto a soma dos valores de *downside* pode ser associado ao risco de uma ilha para a o valor médio da função objetivo.

Note que o valor do logaritmo na Equação (2.1) é menor que zero se a relação entre as médias é menor que 1, caracterizando o *downside* (houve piora de uma geração para outra). Por outro lado, se houve melhora, a razão é maior que 1 e o logaritmo é positivo caracterizando o *upside*. A ideia da medida é relacionar a soma dos *upsides* com a soma dos *downsides*.

No caso deste trabalho, o operador de seleção utilizado é o elitismo, ou seja, a média das soluções nunca piora de uma geração para a seguinte. Por isso, a fórmula do índice de desempenho apenas considera o *upside*, pois não há *downside* neste modelo.

2.2.2 Migração

No momento de aplicar o operador de migração, é selecionada uma parcela dos indivíduos de uma ilha, correspondente a 25% de todos os indivíduos, que vão para uma ilha escolhida utilizando-se um cálculo que envolve aleatoriedade. A ilha-destino é determinada através de um cálculo que envolve o índice de desempenho de cada ilha, ou seja, o quão bem a ilha está se saindo em resolver os problemas em questão. Sempre haverá migração em uma ilha, mesmo se esta tiver apenas 5 indivíduos (neste caso, apenas 1 irá migrar). A topologia escolhida para este trabalho é a de grafo completo (topologia totalmente conectada), como mostra a Figura 2.2. Isso significa que um indivíduo pode ir para qualquer outra ilha, todas são uma possibilidade de destino.



Fonte: autor

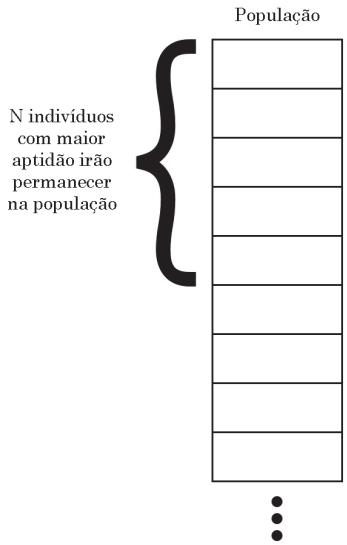
Figura 2.2: Migração com topografia de grafo completo

A migração baseada neste cálculo do índice de desempenho faz com que as ilhas com maior índice fiquem com uma população maior, nos mostrando assim qual é o algoritmo mais eficiente (no contexto da Equação (2.1)) para determinado problema, que será o algoritmo que está sendo utilizado em ilhas com as maiores populações.

2.2.3 Elitismo

Para aplicar o operador de seleção, foi escolhido o elitismo, ou seja, após a avaliação dos indivíduos ser realizada, os N melhores indivíduos serão escolhidos para permanecerem

na população. Os outros serão descartados para dar lugar aos filhos dos indivíduos mais aptos. Essa escolha é justificada pois com o elitismo, temos somente retornos positivos, já que a aptidão média da população sempre diminui no caso de minimização. Caso não houvesse elitismo, a média da população poderia variar para cima e para baixo gerando retornos positivos e negativos. Como o índice de desempenho é baseado apenas no retorno, o índice da ilha nunca irá diminuir de uma geração para a seguinte.



Fonte: autor

Figura 2.3: Seleção por elitismo

2.3 Conjunto de funções-teste

Para verificar se o novo modelo apresenta resultados competitivos, é necessário testá-lo para diferentes problemas, ou funções objetivo. Dentre os vários problemas disponíveis na literatura, foram escolhidos os problemas CEC-2014 (Conference on Evolutionary Computation) [33]. São 30 problemas, todos mono-objetivo de minimização, divididos em 3 tipos: funções básicas, funções híbridas e funções de composição. Um detalhamento do conjunto de funções-teste é apresentado a seguir.

2.3.1 Funções básicas

As funções básicas empregadas para avaliar os modelos desenvolvidos neste trabalho são mostradas na Tabela 2.1.

Tabela 2.1: Funções básicas

Função	Nome	Expressão
F_1	Rotated High Conditioned Elliptic Function	$F_1(x) = f_1(M(x - o_1)) + F_1^*$
F_2	Rotated Bent Cigar Function	$F_2(x) = f_2(M(x - o_2)) + F_2^*$
F_3	Rotated Discus Function	$F_3(x) = f_3(M(x - o_3)) + F_3^*$
F_4	Shifted and Rotated Rosenbrock's Function	$F_4(x) = f_4(M(\frac{2.048(x-o_4)}{100}) + 1) + F_4^*$
F_5	Shifted and Rotated Ackley's Function	$F_5(x) = f_5(M(x - o_5)) + F_8^*$
F_6	Shifted and Rotated Weierstrass Function	$F_6(x) = f_6(M(\frac{0.5(x-o_6)}{100})) + F_6^*$
F_7	Shifted and Rotated Griewank's Function	$F_7(x) = f_7(M(\frac{600(x-o_7)}{100})) + F_7^*$
F_8	Shifted Rastrigin's Function	$F_8(x) = f_8((\frac{5.12(x-o_8)}{100})) + F_7^*$
F_9	Shifted and Rotated Rastrigin's Function	$F_9(x) = f_8(M(\frac{5.12(x-o_9)}{100})) + F_9^*$
F_{10}	Shifted Schwefel's Function	$F_{10}(x) = f_9((\frac{1000(x-o_{10})}{100})) + F_{10}^*$
F_{11}	Shifted and Rotated Schwefel's Function	$F_{11}(x) = f_9(M(\frac{1000(x-o_{11})}{100})) + F_{11}^*$
F_{12}	Shifted and Rotated Katsuura Function	$F_{12}(x) = f_{10}(M(\frac{5(x-o_{12})}{100})) + F_{12}^*$
F_{13}	Shifted and Rotated HappyCat Function	$F_{13}(x) = f_{11}(M(\frac{5(x-o_{13})}{100})) + F_{13}^*$
F_{14}	Shifted and Rotated HGBat Function	$F_{14}(x) = f_{12}(M(\frac{5(x-o_{14})}{100})) + F_{14}^*$
F_{15}	Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function	$F_{15}(x) = f_{13}(M(\frac{5(x-o_{15})}{100}) + 1) + F_{15}^*$
F_{16}	Shifted and Rotated Expanded Scaffer's F6 Function	$F_{16}(x) = f_{14}(M(x - o_{16}) + 1) + F_{16}^*$

2.3.2 Funções hibrídas

As próximas funções são híbridas, nas quais as variáveis são aleatoriamente divididas em subcomponentes, e várias funções básicas são utilizadas para cada subcomponente. Eas funções híbridas tem a forma geral

$$F(x) = g_1(M_1 z_1) + g_2(M_2 z_2) + ... + g_N(M_N z_N) + F^*(x)$$

onde $F(x)$ é a função híbrida, $g_i(x)$ são as funções básicas utilizadas para construir a função híbrida, e N é o número de funções básicas. Também é utilizado um vetor p que define a porcentagem para cada uma das funções básicas $g_i(x)$. A Tabela 2.2 mostra a lista de funções híbridas.

Tabela 2.2: Funções híbridas

Função	Nome	Detalhes
F_{17}	Função híbrida 1	$N = 3$, $p = [0.3, 0.3, 0.4]$, funções F_9, F_8, F_1
F_{18}	Função híbrida 2	$N = 3$, $p = [0.3, 0.3, 0.4]$, funções F_9, F_{12}, F_8
F_{19}	Função híbrida 3	$N = 4$, $p = [0.2, 0.2, 0.3, 0.3]$, funções F_7, F_6, F_4, F_{14}
F_{20}	Função híbrida 4	$N = 4$, $p = [0.2, 0.2, 0.3, 0.3]$, funções F_{12}, F_3, F_{13}, F_8
F_{21}	Função híbrida 5	$N = 5$, $p = [0.1, 0.2, 0.2, 0.2, 0.3]$, funções $F_{14}, F_{12}, F_4, F_9, F_1$
F_{22}	Função híbrida 6	$N = 5$, $p = [0.1, 0.2, 0.2, 0.2, 0.3]$, funções $F_{10}, F_{11}, F_{13}, F_9, F_5$

2.3.3 Funções de composição

As últimas funções são as funções de composição do tipo

$$F(x) = F^* + \sum_{i=1}^{N} w_i * (\lambda_i g_i(x) + bias_i) \tag{2.2}$$

onde $g_i(x)$ são as funções básicas utilizadas nessa composição, N é o número de funções básicas, $bias_i$ é o que define qual ótimo é o ótimo global, λ_i é usado para definir a altura das funções básicas, e w_i define o peso das mesmas funções. w_i é calculado a partir da fórmula

$$w_i = \frac{1}{\sqrt{\sum_{j=1}^{D}(x_j - o_{ij})^2}} exp(-\frac{\sum_{j=1}^{D}(x_j - o_{ij})^2}{2D\sigma_i^2}) \tag{2.3}$$

onde σ é utilizado para controlar o intervalo da função básica $g_i(x)$. A Tabela 2.3 mostra as funções de composição empregadas na avaliação dos modelos usados neste trabalho.

Tabela 2.3: Funções de composição.

Função	Nome	Detalhes
F_{23}	Função de composição 1	$N = 5$, $\sigma = [10, 20, 30, 40, 50]$, $\lambda = [1, 1e{-}6, 1e{-}26, 1e{-}6, 1e{-}6]$, funções F_4, F_1, F_2, F_3, F_1
F_{24}	Função de composição 2	$N = 3$, $\sigma = [20, 20, 20]$, $\lambda = [1, 1, 1]$, funções F_{10}, F_9, F_{14}
F_{25}	Função de composição 3	$N = 3$, $\sigma = [10, 30, 50]$, $\lambda = [0.25, 1, 1e-7]$, funções F_{11}, F_9, F_1
F_{26}	Função de composição 4	$N = 5$, $\sigma = [10, 10, 10, 10, 10]$, $\lambda = [0.25, 1, 1e-7, 2.5, 10]$, funções $F_{11}, F_{13}, F_1, F_6, F_7$
F_{27}	Função de composição 5	$N = 5$, $\sigma = [10, 10, 10, 20, 20]$, $\lambda = [10, 10, 2.5, 25, 1e-6]$, funções $F_{14}, F_9, F_{11}, F_6, F_1$
F_{28}	Função de composição 6	$N = 5$, $\sigma = [10, 20, 30, 40, 50]$, $\lambda = [2.5, 10, 2.5, 5e-4, 1e-6]$, funções $F_{15}, F_{13}, F_{11}, F_{16}, F_1$
F_{29}	Função de composição 7	$N = 3$, $\sigma = [10, 30, 50]$, $\lambda = [1, 1, 1]$, funções F_{17}, F_{18}, F_{19}
F_{30}	Função de composição 8	$N = 3$, $\sigma = [10, 30, 50]$, $\lambda = [1, 1, 1]$, funções F_{20}, F_{21}, F_{22}

2.4 Algoritmos utilizados

Neste modelo, cada ilha terá um algoritmo evolutivo diferente, cada um com suas características próprias no que diz respeito à diversidade das soluções, profundidade, etc. O esperado do modelo é que cada um desses algoritmos percorra um caminho diferente no espaço de soluções, contribuindo entre si através da realização das migrações. A biblioteca utilizada possui diversos algoritmos para serem escolhidos, tanto mono-objetivo como multi-objetivo. No nosso caso, apenas algoritmos que lidam com problemas mono-objetivo foram escolhidos. Os algoritmos escolhidos foram o DE (Differential Evolution), proposto por Storn e Price (1995), SGA (Simple Genetic Algorithm), HS (Harmony Search), proposto por Geem (2001) e ES (Evolution Strategy), proposto por Rechemberg e Schwefel (1970).

2.4.1 Algoritmo SGA

O SGA (Simple Genetic Algorithm) [7] possui uma população de soluções, e uma função objetivo a ser otimizada. Cada solução (indivíduo) possui sua fitness, que representa o quão apto o indivíduo é em resolver o determinado problema.

No momento da inicialização é gerada uma população de indivíduos, cujo tamanho depende do problema a ser otimizado. O genótipo desses individos é gerado aleatoriamente, dentro do intervalo do espaço de soluções. Em alguns casos, os indivíduos iniciais tem mais chance de serem criados em áreas onde é mais provável que a solução ótima esteja. Cada indivíduo pode ser representado por exemplo por um vetor de bits, onde cada bit ou conjunto de bits representa alguma característica do indivíduo.

Na parte da seleção, uma parte da população é selecionada para gerar novos indivíduos. Geralmente, os indivíduos a serem selecionados são os que possuem as melhores fitness para o determinado problema.

Para gerar a próxima geração de invidíduos da população, é necessário aplicar os chamados operadores genéticos, como crossover e mutação. Para cada indivíduo a ser criado, são obtidas características de 2 outros indivíduos da população (os pais) e assim é gerado o genótipo do novo indivíduo. Este indivíduo está sujeito a mutações, nas quais o genótipo do indivíduo possui uma chance de ser alterado de alguma maneira.

O algoritmo finaliza quando o critério de parada (definido no início) for alcançado.

Esse critério de parada pode ser de vários tipos, como pro exemplo uma solução boa o suficiente, um número de gerações máximo, entre outras.

2.4.2 Algoritmo DE

Na computação evolutiva, o algoritmo DE (Differential Evolution) [16] é um algoritmo que busca melhorar uma solução por meio de otimizá-la iterativamente. É um método que não se utiliza do gradiente do problema a ser otimizado, portanto funciona para problemas que não são contínuos. O algoritmo DE pode ser descrito nos seguintes passos:

1. Iniciar todos os indivíduos da população em posições aleatórias no espaço de busca.
2. Enquanto o critério de parada não for satisfeito, repetir os seguintes passos:
   (a) Para cada indivíduo na população, faça:
      i. Escolher 3 outros indivíduos na população.
      ii. Produzir um vetor utilizando-se combinações de características das 3 soluções escolhidas no passo anterior.
      iii. Substituir o indivíduo pelo novo indivíduo criado, caso este seja uma solução melhor. Caso contrário, manter o indivíduo.

2.4.3 Algoritmo ES

O algoritmo ES (Evolution Strategy) [18] usa a mutação e seleção como operadores de busca. Assim como os outros algoritmos evolutivos, os operadores são aplicados em um laço, até que um critério de parada seja alcançado. Neste algoritmo, a seleção é baseada apenas no ranking da fitness dos indivíduos, e não em seus valores. Assim, os indivíduos que estiverem nas posições altas do ranking serão selecionados e irão gerar novos indivíduos para a próxima geração, enquanto os outros serão descartados.

2.4.4 Algoritmo HS

O algoritmo HS (Harmony Search) [19] é um algoritmo bem simples, porém bastante eficiente. O início do algoritmo é o mesmo de vários algoritmos genéticos conhecidos, tendo uma população gerada aleatoriamente no espaço de soluções. Uma grande peculiaridade do algoritmo é que no momento de gerar uma nova solução, essa nova solução é gerada

baseando-se em todas as soluções na memória do algoritmo, ao invés de besear-se em 2 indivíduos pais, como na maioria dos algoritmos. Essa memória é construída com base em observações de momentos nos quais as soluções produziram bons resultados.

3 EXPERIMENTOS COMPUTACIONAIS E DISCUSSÃO

Para executas as simulações deste novo modelo, foi escolhido que o tamanho da população será de 100 indivíduos, divididos igualmente entre as ilhas no início da simulação. Os parâmetros dos algoritmos utilizados em cada ilha foram são mostrados na Tabela 3.1.

Tabela 3.1: Parâmetros dos algoritmos.

Algoritmo	Nome	Parâmetros
DE	Differential Evolution	coeficiente de peso F = 0.8, probabilidade de crossover = 0.9, variante de mutação = DE/rand/1/exp, critério de parada na tolerância tol = 1e-6, critério de parada na tolerância x = 1e-6
SGA	Simple Genetic Algorithm	probabilidade de crossover = 0.95, probabilidade de mutação = 0.05, estratégia de seleção = "truncated", semente utilizada pelo gerador de números aleatórios = 0
HS	Harmony Search	taxa de escolha da memória = 0.85, taxa mínima de ajuste de afinação = 0.35, taxa máxima de ajuste de afinação = 0.99, distância mínima da largura de banda = 1e-5, distância máxima de largura de banda = 1
ES	Evolution Strategies	passo inicial (σ) = 0.5. Os outros parâmetros do algoritmo são calculados automaticamente ao longo das gerações.

Durante a realização das simulações do modelo de ilhas, os melhores valores da função objetivo foram salvas para cada ilha a cada geração, para que ao fim se possa comparar os resultados.

3.1 Primeira etapa de simulações

Primeiramente, foram simulados modelos com apenas uma ilha, ou seja, apenas um algoritmo evolutivo sendo executado em uma população única de 100 indivíduos. Nestes cenários, foram usados os 4 algoritmos apresentados anteriormente. Os cenários são mostrados na Tabela 3.2.

Tabela 3.2: Composição dos cenários 1 a 4. Os parâmetros usados nos algoritmos são mostrados na Tabela 3.1.

Cenário	Algoritmos utilizados	Tamanho da população	Número de gerações
C-01	Algoritmo DE	100	100
C-02	Algoritmo SGA	100	100
C-03	Algoritmo HS	100	100
C-04	Algoritmo ES	100	100

Esses cenários foram executados para cada um dos 30 problemas do conjunto CEC-2014, para que possamos verificar a qualidade dos resultados e assim comparar quais foram mais eficientes em resolver os problemas. A Tabela 3.3 apresenta os algoritmos mais eficientes para cada um dos problemas. Estas simulações foram executadas com 100 gerações e uma população total de 100 indivíduos.

3.2 Segunda etapa de simulações

Para a segunda etapa de simulações, o modelo proposto será comparado com o melhor algoritmo de cada um dos 30 problemas, para que se possa medir o desempenho do mesmo. Os cenários testados são mostrados na Tabela 3.4. Nestes cenários, foram usados os 4 algoritmos no cenário C-05 e 3 algoritmos no cenário C-06. A escolha dos algoritmos no cenário C-06 é explicada posteriormente.

A Tabela 3.5 apresenta os resultados médios obtidos em 30 execuções. As simulações foram executadas com 100 gerações e uma população total de 100 indivíduos, divididos entre 5 ilhas com intervalo de migração igual a 10 gerações.

Como pode-se perceber, o modelo proposto obteve soluções similares do que o melhor algoritmo para a maioria dos 30 problemas testados. Para comprovar esses resultados, foi realizado o teste de Tukey para descrobrir se os resultados do melhor algoritmo e do modelo proposto possuem diferenças significativas entre si. O teste foi realizado da seguinte

Tabela 3.3: Melhores algoritmos por problema para a primeira etapa de simulações. As simulações foram executadas com 100 gerações e uma população total de 100 indivíduos.

Problema	MelhorAlgoritmo	Ótimo	Solução	Desvio padrão
F_1	DE	100	248.61	39.63
F_2	DE	200	200.49	0.18
F_3	DE	300	300.00	1.57
F_4	DE	400	400.00	0.0007
F_5	DE	500	519.92	0.61
F_6	DE	600	600.11	0.08
F_7	HS	700	700.08	0.07
F_8	DE	800	800.00	8.89
F_9	HS	900	905.93	2.18
F_{10}	SGA	1000	1000.07	0.03
F_{11}	HS	1100	1190.19	46.38
F_{12}	ES	1200	1200.11	0.03
F_{13}	HS	1300	1300.15	0.03
F_{14}	DE	1400	1400.15	0.02
F_{15}	HS	1500	1500.13	0.20
F_{16}	HS	1600	1601.25	0.32
F_{17}	DE	1700	1750.45	11.15
F_{18}	DE	1800	1805.15	0.88
F_{19}	ES	1900	1900.73	0.37
F_{20}	DE	2000	2002.55	0.41
F_{21}	DE	2100	2103.29	1.07
F_{22}	HS	2200	2200.76	0.66
F_{23}	DE	2300	2629.46	1.39
F_{24}	HS	2400	2515.43	3.90
F_{25}	DE	2500	2637.34	3.27
F_{26}	HS	2600	2700.14	0.03
F_{27}	DE	2700	2703.30	0.31
F_{28}	DE	2800	3156.88	0.05
F_{29}	DE	2900	3061.34	18.92
F_{30}	DE	3000	3576.19	30.58

maneira: para cada um dos 30 problemas, verificou-se qual dos algoritmos utilizados obteve a melhor solução na simulação. Em seguida, foi feito um teste de Tukey entre as soluções desse melhor algoritmo com as soluções do modelo proposto, para verificar se há uma diferença significativa entre os dois modelos. Como pode-se ver na Tabela 3.5, não houve diferença significativa em nenhum dos 30 problemas utilizados.

Tabela 3.4: Composição dos cenários 5 e 6. Os parâmetros usados nos algoritmos são mostrados na Tabela 3.1.

Cenário	Algoritmos utilizados	Tamanho da população	Número de gerações	Frequência de migração
C-05	Algoritmos DE, SGA, ES e HS	100 (divididos em 4 ilhas com 25 indivíduos)	100	10 gerações
C-06	Algoritmos SGA, ES e HS	100 (divididos em 2 ilhas com 33 indivíduos e 1 ilha com 34 indivíduos)	100	10 gerações

3.3 Análise dos resultados com o método performance profiles

Para fazer uma comparação entre os diferentes cenários utilizados neste trabalho, foi utilizada uma técnica denominada Performance Profiles, explicada em [34]. Essa técnica é aplicável em avaliações em que se tem um conjunto de algoritmos e outro de problemas. É possível avaliar e comparar o desempenho do conjunto de algoritmos, na resolução do conjunto de problemas [31].

O teste gira em torno do cálculo de $\rho_s(\tau)$, que corresponde a quantos problemas o algoritmo s consegue resolver sob o custo τ, e assim nos mostra qual é o algoritmo mais eficiente em resolver o determinado problema. Este teste mostra detalhadamente quais algoritmos convergem mais rapidamente e quais apresentam as melhores soluções. Para realizar o teste, é necessário seguir os seguintes passos:

Seja $t_{p,s}$ o tempo necessário para resolver o problema p com o algoritmo s. Calcula-se então o valor de $r_{p,s}$, que é definido como:

$$r_{p,s} = \frac{t_{p,s}}{min\{t_{p,s} : s \in S\}} \tag{3.1}$$

onde S é o conjunto de todos os algoritmos disponíveis. Além disso, é necessário declarar r_M, que é um valor maior ou igual a rp, s para todo $r_{p,s}$.
Então, finalmente calcula-se o valor de $\rho_s(\tau)$, que fornece quantos problemas o algoritmo s resolve sob o custo τ:

$$\rho_s(\tau) = \frac{1}{N}|p \in P : r_{p,s} \leq \tau| \tag{3.2}$$

onde N é o número de problemas disponíveis.

Tabela 3.5: Comparação do melhor algoritmo com o modelo proposto (MP). A segunda coluna apresenta o algoritmo que apresenta o melhor resultado médio em 30 execuções com uma população de 100 indivíduos em 100 gerações. As simulações com o modelo proposto nessa dissertação (MP) foram executadas com 100 gerações e uma população total de 100 indivíduos, divididos entre 5 ilhas com intervalo de migração de 10 gerações. O teste de Tukey foi executado para verificar se há diferenças significativas.

Problema	Melhor Algoritmo	Ótimo	Solução	Desvio padrão	Solução (MP)	D.P. (MP)	Possui diferença signific.
F_1	DE	100	248.61	39.63	118.12	36.85	Não
F_2	DE	200	200.49	0.18	200.02	0.02	Não
F_3	DE	300	300.00	1.57	300.00	0.00	Não
F_4	DE	400	400.00	0.0007	403.48	10.43	Não
F_5	DE	500	519.92	0.61	520.00	0.0002	Não
F_6	DE	600	600.11	0.08	600.10	0.27	Não
F_7	HS	700	700.08	0.07	700.08	0.05	Não
F_8	DE	800	800.00	8.89	800.00	0.00	Não
F_9	HS	900	905.93	2.18	908.58	2.17	Não
F_{10}	SGA	1000	1000.07	0.03	1000.03	0.04	Não
F_{11}	HS	1100	1190.19	46.38	1226.09	107.42	Não
F_{12}	ES	1200	1200.11	0.03	1200.02	0.02	Não
F_{13}	HS	1300	1300.15	0.03	1300.12	0.02	Não
F_{14}	DE	1400	1400.15	0.02	1400.16	0.06	Não
F_{15}	HS	1500	1500.13	0.20	1501.13	0.53	Não
F_{16}	HS	1600	1601.25	0.32	1601.70	0.59	Não
F_{17}	DE	1700	1750.45	11.15	1719.56	25.17	Não
F_{18}	DE	1800	1805.15	0.88	1803.25	2.42	Não
F_{19}	ES	1900	1900.73	0.37	1900.43	0.34	Não
F_{20}	DE	2000	2002.55	0.41	2001.69	0.97	Não
F_{21}	DE	2100	2103.29	1.07	2105.85	8.02	Não
F_{22}	HS	2200	2200.76	0.66	2201.63	3.82	Não
F_{23}	DE	2300	2629.46	1.39	2629.46	0.00	Não
F_{24}	HS	2400	2515.43	3.90	2515.59	4.66	Não
F_{25}	DE	2500	2637.34	3.27	2635.30	7.29	Não
F_{26}	HS	2600	2700.14	0.03	2700.10	0.02	Não
F_{27}	DE	2700	2703.30	0.31	2792.58	136.68	Não
F_{28}	DE	2800	3156.88	0.05	3141.54	81.22	Não
F_{29}	DE	2900	3061.34	18.92	3117.90	24.66	Não
F_{30}	DE	3000	3576.19	30.58	3534.11	36.91	Não

Para realizar a comparação, foram utilizados 6 cenários, sendo 4 deles modelos com apenas uma população única, um quinto cenário com o modelo de ilhas proposto, e um sexto cenário com uma variação nos parâmetros do modelo proposto.

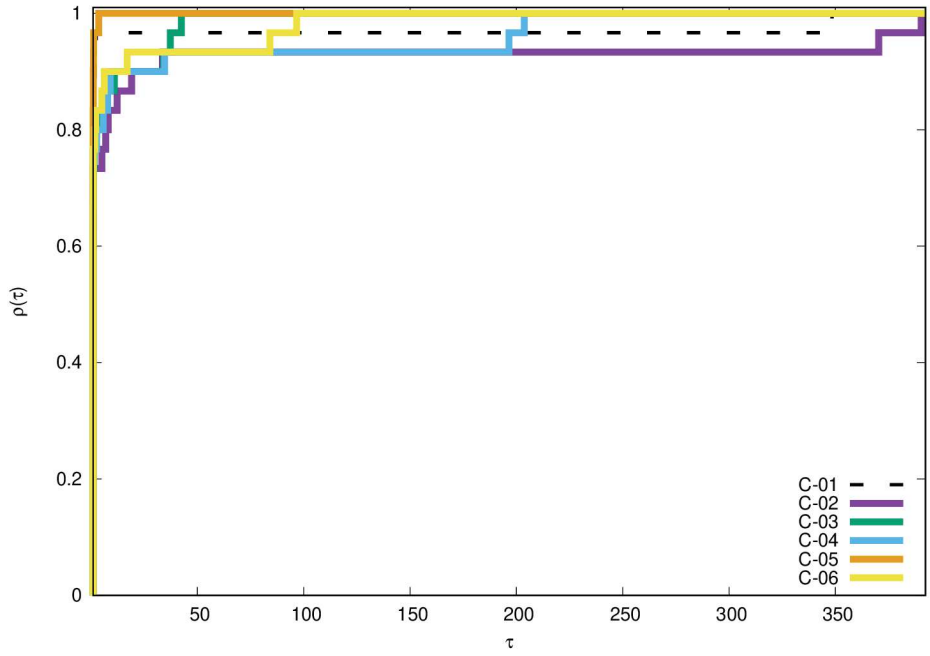
1. C-01: Algoritmo DE
2. C-02: Algoritmo SGA
3. C-03: Algoritmo HS
4. C-04: Algoritmo ES
5. C-05: Modelo proposto
6. C-06: Modelo proposto sem o algoritmo DE

Para uma análise mais completa da capacidade de adaptação do modelo proposto, foi criado o sexto cenário que utiliza o modelo proposto sem o algoritmo DE, ou seja, com apenas 3 ilhas contendo os algoritmos SGA, HS e ES. Isso foi feito porque o algoritmo DE é considerado muito eficiente em comparação com os outros algoritmos utilizados e contribui fortemente para a solução da maioria dos prolemas analisados. Logo, é necessário avaliar como o novo modelo proposto se adapta sem a utilização deste algoritmo. Além disso, foram feitas simulações com 100, 500 e 1000 gerações, para verificar se os resultados apresentam variações com diferentes números de gerações.

A Figura 3.1 apresenta os resultados dos perfis de desempenho para a execução do conjunto de problemas ao longo de 100 gerações em 30 execuções independentes. O intervalo de migrações foi ajustado para 10 gerações. Como podemos observar na Figura 3.1, o cenário C-05 apresentou o melhor desempenho pois é o cenário do novo modelo utilizando todos os algoritmos, ou seja, é esperado que ele tenha o melhor resultado pois ele utiliza as vantagens de todos os algoritmos que o compõe. Ele é o melhor cenário seguido do cenáro C-03 (HS) e C-06 (novo modelo sem o DE). O cenário que apresentou o pior desempenho para o conjunto de simulações realizadas foi o cenário C-02 (SGA) em função de não ser um algoritmo eficiente para os tipos de problemas escolhidos.

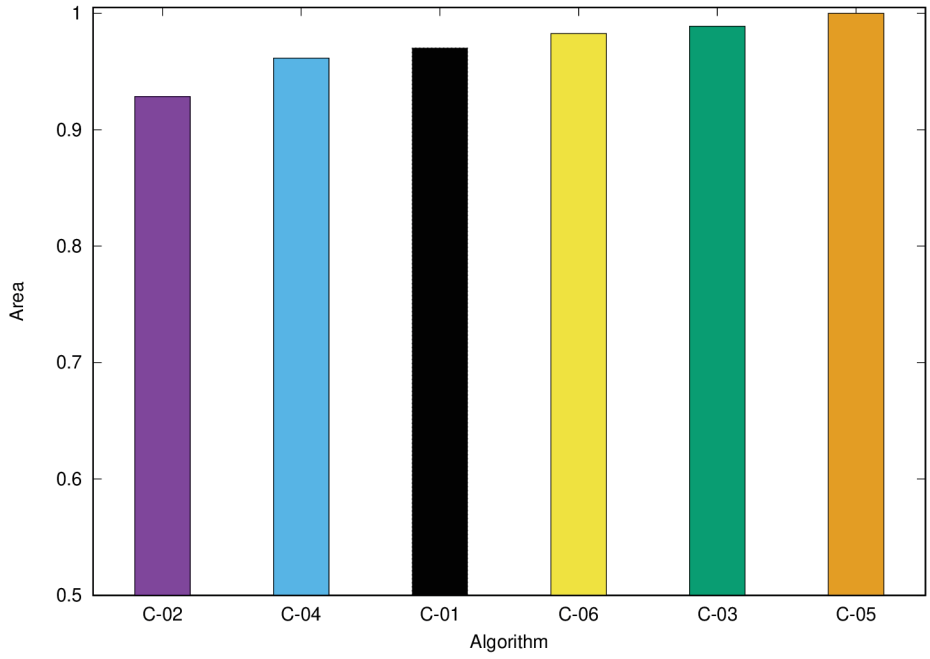
Mesmo não utilizando o algoritmo DE, o cenário C-06 conseguiu encontrar o valor $\rho(\tau) = 1$. Isso mostra que o modelo proposto no cenário debilitado (sem o algoritmo DE) apresenta soluções igualmente boas às melhores soluções dos outros cenários. O cenário C-06 é o melhor após o C-05 (modelo com todos os 4 algoritmos) e o C-03.

Os resultados da área sobre a curva são mostrados na Figura 3.2. Ao fazer uma análise do gráfico, pode-se observar algumas informações importantes, como por exemplo, qual cenário que oferece um melhor resultado para a maioria dos problemas propostos, qual



Fonte: autor

Figura 3.1: Perfis de desempenho para a simulação dos 30 problemas, utilizando-se os 6 cenários com 100 gerações em 30 execuções independentes. O intervalo de migrações foi ajustado para 10 gerações.



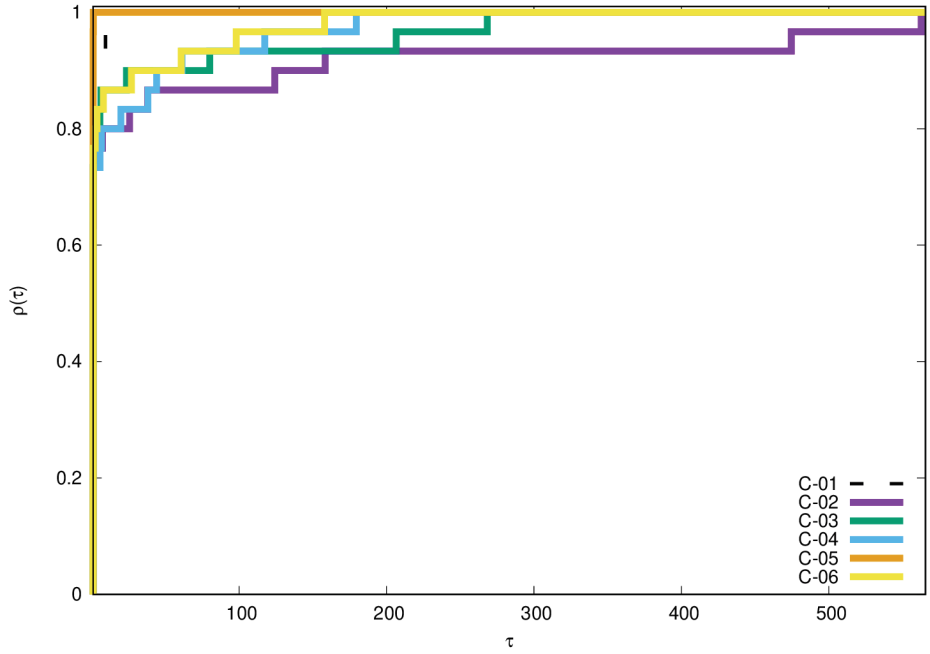
Fonte: autor

Figura 3.2: Gráfico de barras mostrando a área sobre a curva para a simulação dos 30 problemas, utilizando-se os 6 cenários com 100 gerações. O intervalo de migrações foi ajustado para 10 gerações.

cenário é mais robusto, ou seja, qual apresenta o menor valor de τ tal que $\rho(\tau) = 1$, e também qual cenário possui a maior área embaixo da curva (melhores soluções). Podemos perceber mais claramente nessa figura que o cenário C-06 ficou em terceiro melhor, mas

muito próximo do melhor cenário.

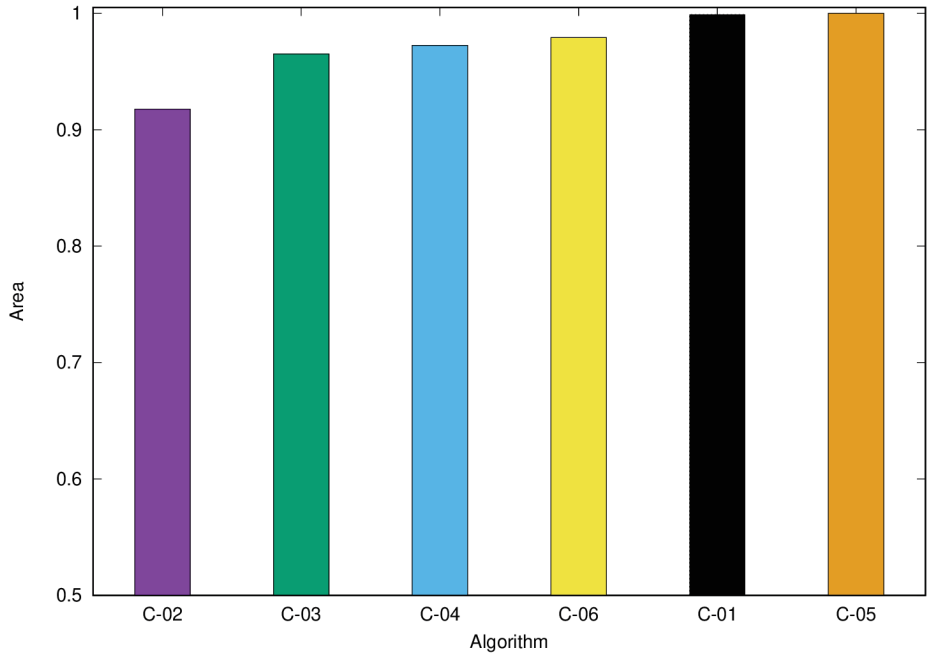
A Figura 3.3 apresenta os resultados dos perfis de desempenho para a execução do conjunto de problemas ao longo de 500 gerações em 30 execuções independentes. Como podemos observar na Figura 3.3, o cenário C-05 também apresentou o melhor desempenho, assim como nas simulações de 100 gerações. Ele é o melhor cenário seguido do cenáro C-01 (DE) e C-06 (modelo proposto sem o DE). O cenário que apresentou o pior desempenho para o conjunto de simulações realizadas foi o cenário C-02 (SGA). Os resultados da área sobre a curva são mostrados na Figura 3.4. Nesta figura o cenário C-06 também ficou em terceiro melhor.



Fonte: autor

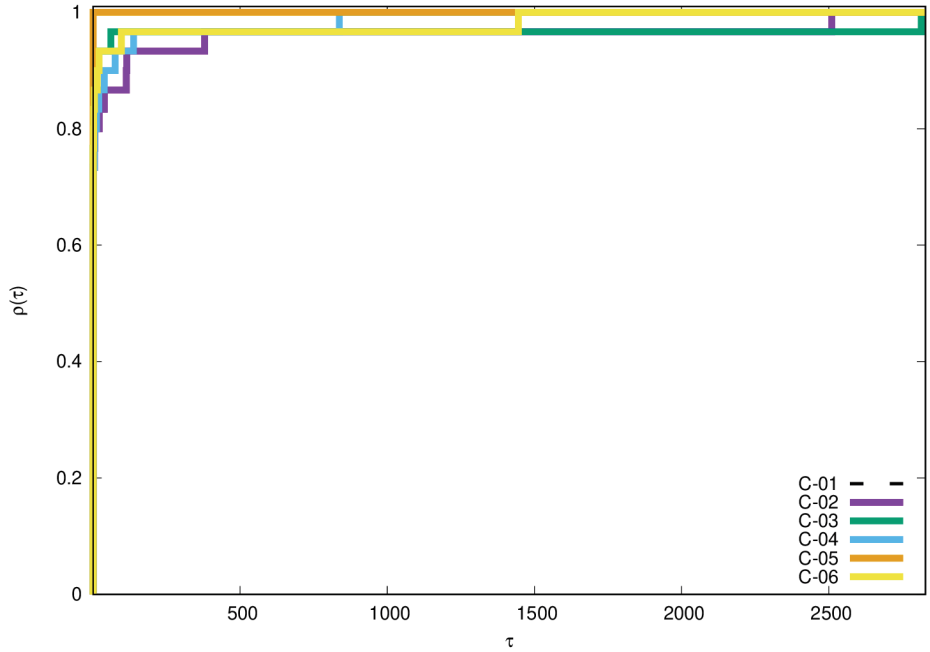
Figura 3.3: Perfis de desempenho para a simulação dos 30 problemas, utilizando-se os 6 cenários com 500 gerações.

A Figura 3.5 apresenta os resultados dos perfis de desempenho para a execução do conjunto de problemas ao longo de 1000 gerações em 30 execuções independentes. Os resultados da área sobre a curva são mostrados na Figura 3.6. Como podemos observar na Figura 3.5, o cenário C-05 também apresentou o melhor desempenho, assim como em todas as outras simulações. Ele é o melhor cenário seguido do cenáro C-01 (DE) e C-04 (ES). O cenário que apresentou o pior desempenho para o conjunto de simulações realizadas foi novamente o cenário C-02 (SGA).



Fonte: autor

Figura 3.4: Gráfico de barras mostrando a área sobre a curva para a simulação dos 30 problemas, utilizando-se os 6 cenários com 500 gerações.

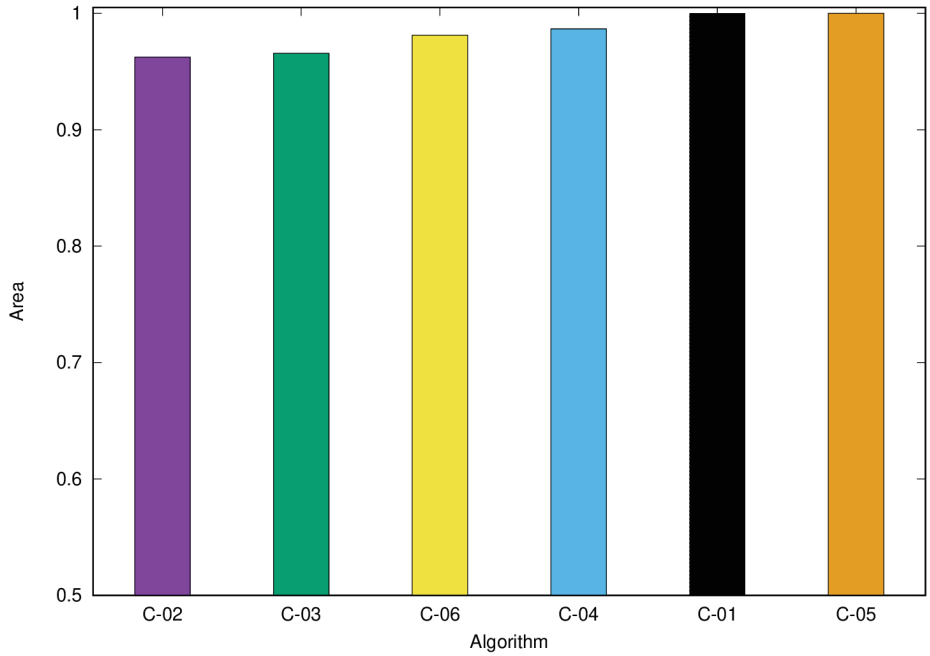


Fonte: autor

Figura 3.5: Perfis de desempenho para a simulação dos 30 problemas, utilizando-se os 6 cenários com 1000 gerações.

3.4 Análise com dados extraídos das simulações

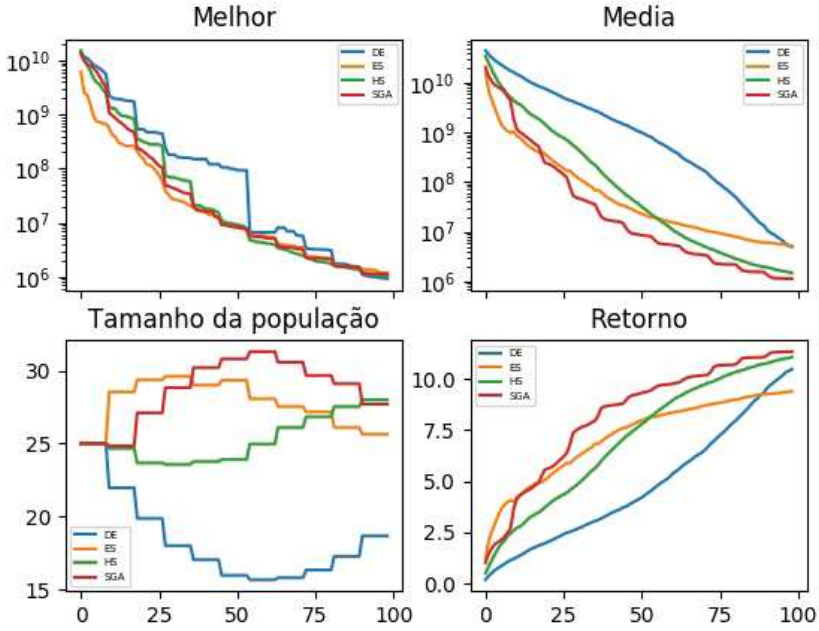
Durante a execução do modelo, algumas informações foram salvas, informações que dizem respeito ao que está acontecendo com os indivíduos das ilhas. Essas informações



Fonte: autor

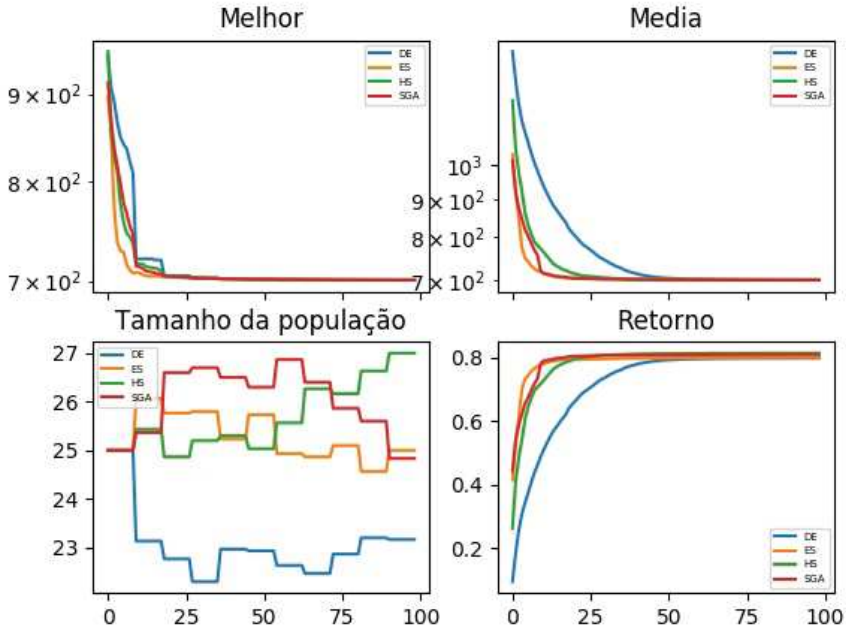
Figura 3.6: Gráfico de barras mostrando a área sobre a curva para a simulação dos 30 problemas, utilizando-se os 6 cenários com 1000 gerações.

são: o melhor indivíduo, a média das soluções, o tamanho da população e o retorno de cada uma das ilhas do modelo. Essas informações são úteis pois elas auxiliam nas conclusões sobre o porquê de a simulação apresentar certos resultados. As Figuras 3.7, 3.8, 3.9 e 3.10 mostram a variação destes dados ao longo da execução do cenário C-05 para 4 dos problemas utilizados. Esses problemas foram escolhidos porque em cada um deles, um algoritmo diferente apresentou a melhor solução, como pode ser obervado na Tabela 3.5.



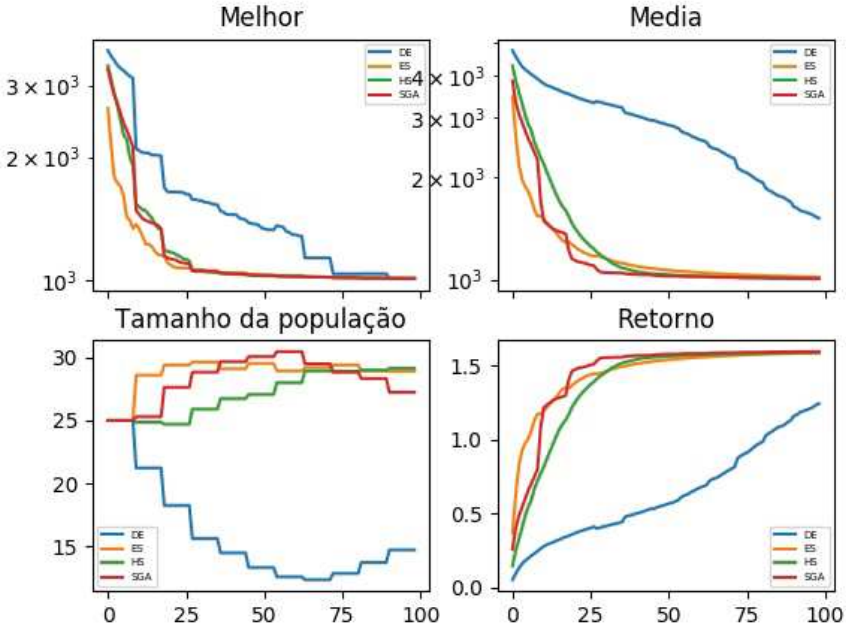
Fonte: autor

Figura 3.7: Detalhes da simulação para a função F_2 (Cenário C-05)



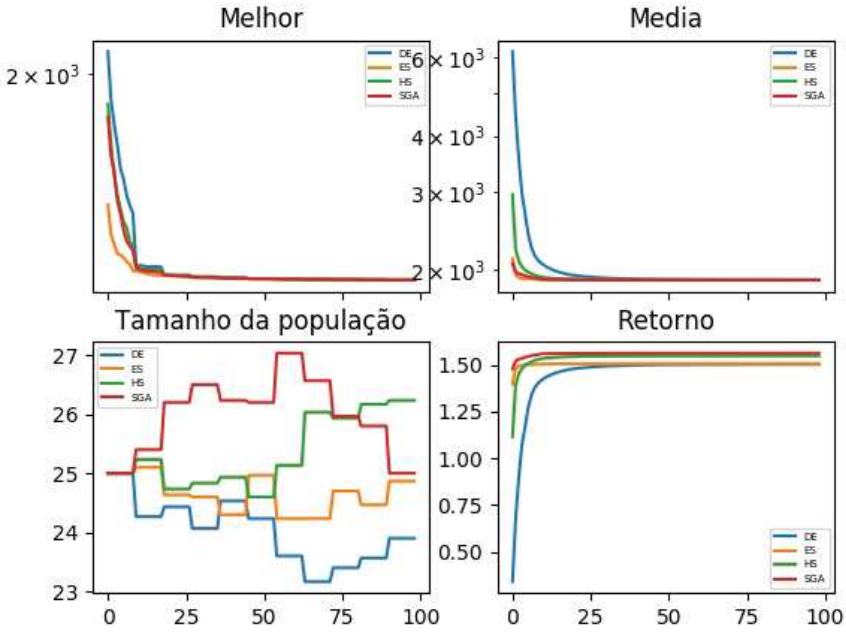
Fonte: autor

Figura 3.8: Detalhes da simulação para a função F_7 (Cenário C-05)



Fonte: autor

Figura 3.9: Detalhes da simulação para a função F_{10} (Cenário C-05)



Fonte: autor

Figura 3.10: Detalhes da simulação para a função F_{19} (Cenário C-05)

Como podemos perceber nas Figuras 3.7, 3.8, 3.9 e 3.10, a população da ilha que contém o algoritmo DE diminui para estes problemas. Isso ocorre porque o algoritmo

DE gera soluções muito boas logo no início da simulação. Por isso, ele apresenta uma convergência muito rápida e ele deixa de ser atrativo após poucas gerações se passarem. Assim, ele envia soluções boas para as outras ilhas no início do processo evolutivo, fazendo-as ficarem atrativas, e sua população começa a diminuir em seguida. Os outros algoritmos basicamente se beneficiam do DE em detrimento deste.

A Equação (2.1) se utiliza da melhora do resultado de uma geração para a seguinte, ou seja, caso um algoritmo chegue no resultado final já nas primeiras gerações e fique estagnado depois, o retorno da ilha após um certo número de gerações será bem pequeno, pois a fórmula leva em consideração o somatório do logaritmo da divisão da média da população pela média da geração seguinte, para todas as gerações. Com as médias iguais, o valor que será somado à formula do retorno passará a ser igual ao logaritmo de 1, o que vai fazer com que o retorno diminua aos poucos. Vale lembrar que o operador de seleção utilizado é o elitismo, ou seja, a média nunca vai piorar de uma geração para a seguinte. Portanto, caso não haja uma melhora na média das soluções de uma determinada ilha, a média vai estagnar, pois os indivíduos piores serão descartados. Por isso, uma ilha que estagnou no início do processo evolutivo será considerada pior do que uma outra ilha que possui uma média muito pior, mas que ainda não estagnou.

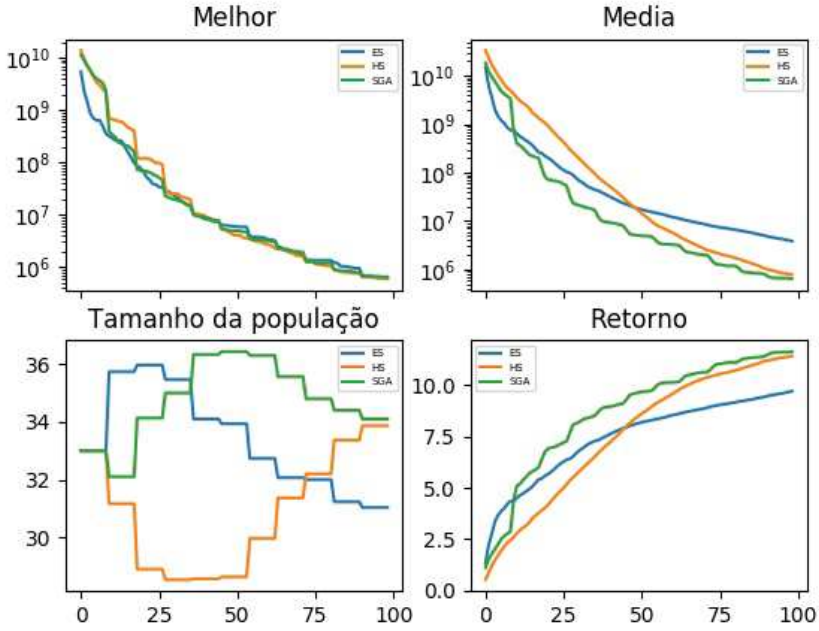
Suponha que em determinada geração, uma das ilhas já tenha chegado a uma solução muito boa e estagnado, e outra ilha tenha soluções piores mas ainda não estagnou. Na próxima geração, a melhoria em relação à geração anterior será maior para a ilha que possui a solução pior, pois ela teve uma melhora percentual melhor (por exemplo, a ilha pior melhorou de 100 para 70, enquanto que a melhor ilha melhorou de 10 para 9). Quando essa informação entrar no somatório da fórmula do retorno, o somatório da pior ilha ficará aos poucos maior do que o da melhor ilha, fazendo com que esta fique pouco atrativa.

Através da análise do comportamento do algoritmo ao longo da simulação (tamanho da população, média das soluções), é possível perceber que o melhor algoritmo para algum problema, geralmente o DE, recebe soluções ruins e exporta soluções boas para as outras ilhas.Isso sugere que o modelo proposto os algoritmos se beneficiem do bom desempenho do melhor algoritmo. Como o DE está sempre melhorando as soluções de baixa qualidade provenientes das outras ilhas, a média dele não parece ser muito boa comparado com a

média das outras ilhas, porém é errado tirar a conclusão de ele é um algoritmo pouco eficiente por causa disso. Ele simplesmente está ajudando as outras ilhas em detrimento de sua atratividade.

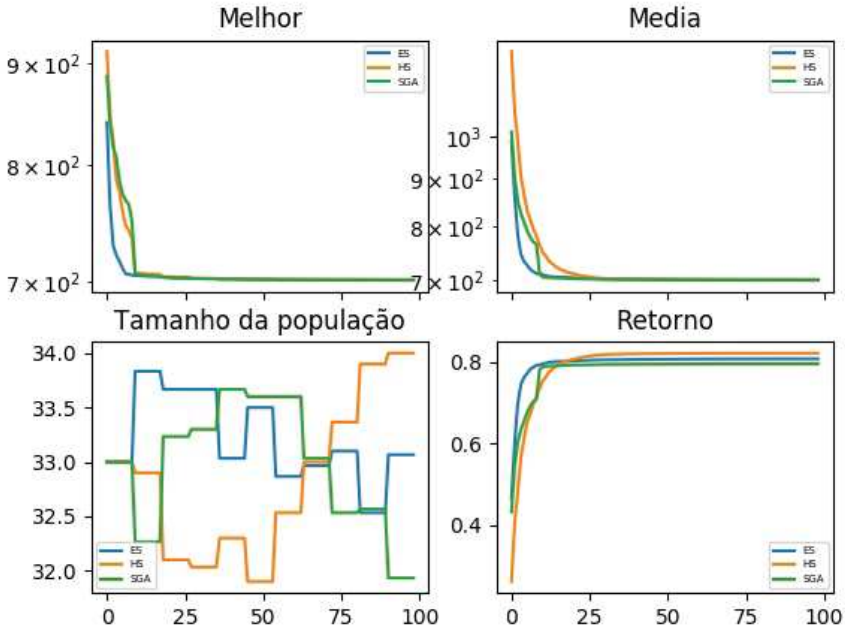
Nós podemos perceber que os valores da média das ilhas são muito parecidos ao final das gerações, o que mostra que o algoritmo que apresentou o melhor desempenho realmente consegue melhorar bastante os indivíduos que chegam em sua ilha. Dessa forma, se estabelece um ciclo na ilha do melhor algoritmo: indivíduos de má qualidade chegando na ilha e indivíduos de boa qualidade sendo enviados para as outras ilhas com maior retorno. Na introdução deste trabalho, foi dito que ao fim da simulação, a ilha que estivesse com o maior número de indivíduos poderia ser a ilha com o algoritmo mais eficiente, pois este recebe mais indivíduos. Porém, agora foi visto que nem sempre é isso que acontece, pois as vezes a ilha que possui o algoritmo mais eficiente fica pouco atrativa para que possa dar indivíduos para as outras ilhas, enquanto recebe indivíduos piores na migração.

As Figuras 3.7, 3.8, 3.9 e 3.10 mostram os mesmos dados da simulação para o cenário C-06. Nesse cenário, o algoritmo que teve sua população mais reduzida foi o HS, assim como o algoritmo DE no cenário C-05. Percebe-se que o melhor indivíduo em cada geração é bem similar entre os 3 algoritmos. Um comportamento similar é observado para a média das soluções, o que indica que a migração entre as ilhas fez com que nenhuma das populações fique com soluções muito distantes das soluções das demais ilhas. Com isso, as qualidades de todos os algoritmos são utilizadas em conjunto para que haja uma grande eficiência no modelo, e assim gerando soluções muito eficientes comparadas com os outros cenários.



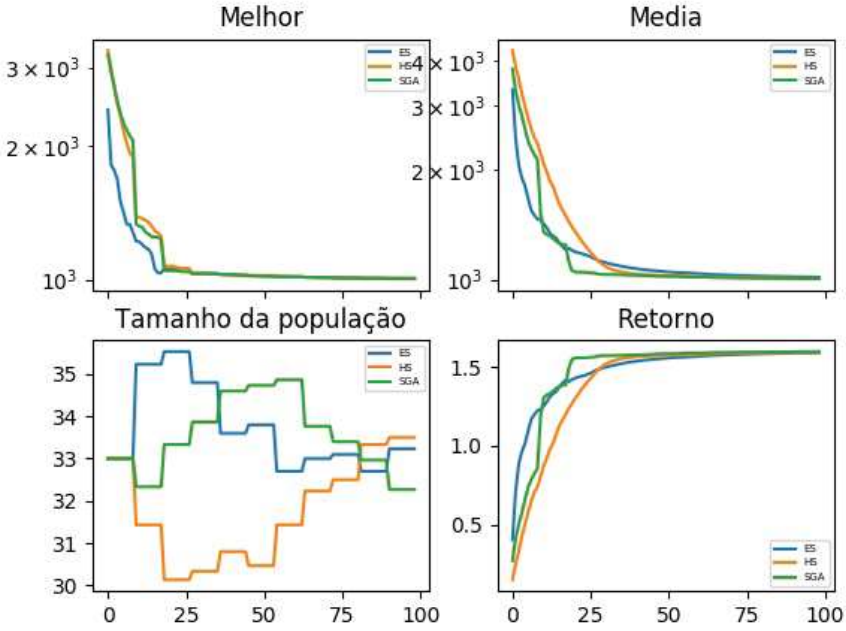
Fonte: autor

Figura 3.11: Detalhes da simulação para a função F_2 (Cenário C-06)



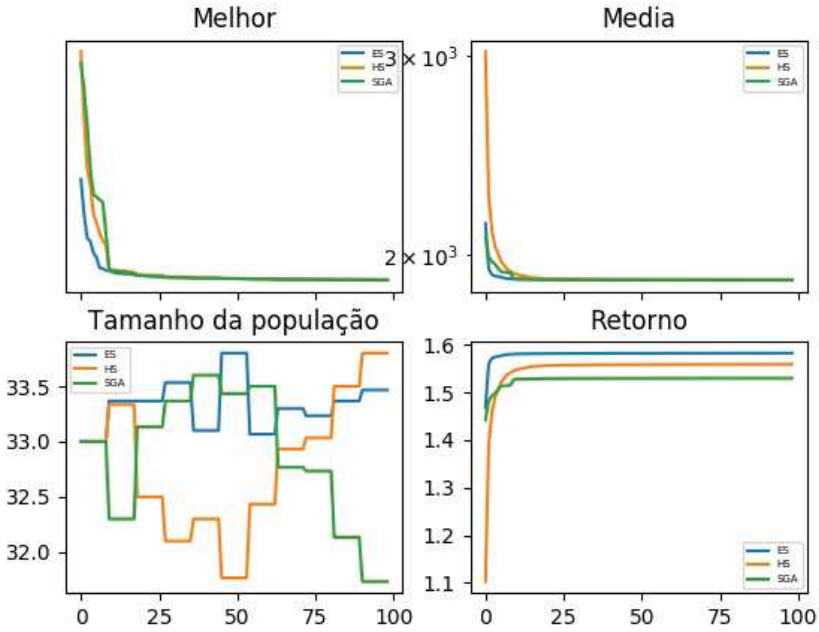
Fonte: autor

Figura 3.12: Detalhes da simulação para a função F_7 (Cenário C-06)



Fonte: autor

Figura 3.13: Detalhes da simulação para a função F_{10} (Cenário C-06)



Fonte: autor

Figura 3.14: Detalhes da simulação para a função F_{19} (Cenário C-06)

Um ponto importante na implementação do modelo é que a diversidade dos algoritmos é essencial para a qualidade das soluções finais. Isso ocorre porque cada um dos algoritmos

contribui com seus pontos positivos. Alguns deles contribuem pelo fato de utilizarem uma abordagem de intensificação, outros contribuem pelo fato de serem mais exploradores. No contexto do modelo proposto, observamos que o algoritmo DE possui uma característica de diminuir sua população e intensificar as soluções. Por isso que ao retirar o algoritmo no cenário C-06, as soluções pioram em qualidade, devido ao fato do modelo perder um algoritmo com uma forte intensificação. Mesmo assim, o cenário C-06 produziu soluções eficientes, pois o modelo possui as vantagens de 3 outros algoritmos que possuem características diferentes entre si, o que contribui para a qualidade da solução final. Mesmo perdendo um algoritmo eficiente, o modelo se adaptou e fez uso das qualidades dos outros algoritmos para manter uma solução eficiente.

4 CONCLUSÃO

Como o modelo proposto apresenta soluções semelhantes ao melhor dos algoritmos utilizados para os 30 problemas analisados, pode-se concluir que este modelo é uma boa opção para grande parte dos problemas, pois ele pode ser utilizado para algum problema genérico e provavelmente irá apresentar soluções semelhantes ao melhor algoritmo, dado que esse foi o caso para os 30 problemas utilizados nesse trabalho. Caso este modelo não seja utilizado, teria-se que testar os algoritmos um a um até encontrar o melhor deles, o que demandaria mais esforço.

Os objetivos do trabalho foram atingidos, pois o modelo proposto apresentou ótimas soluções em comparação com os outros cenários apresentados. Essa conclusão foi comprovada através dos teste dos perfis de desempenho e também através de testes estatísticos.

Os trabalhos futuros observados após os experimentos realizados foram os seguintes:

- Conduzir um estudo paramétrico exaustivo na proposta apresentada, principalmente no número de migrações e na frequência de migração entre as ilhas (algoritmos).
- Incluir uma janela para o cálculo do retorno considerando somente uma determinada quantidade de gerações anteriores ao invés de todo o histórico do processo evolutivo.
- Propor alguma medida de risco associada ao desenvolvimento dos algoritmos, como por exemplo, associar uma medida de risco estrutural mínimo, onde a não melhora da média da função objetivo se configura como um risco para a evolução
- Incluir na proposta acima o conceito de taxa livre de risco, comparativamente ao desempenho de um algoritmo de desempenho mínimo, como por exemplo, o algoritmo de busca aleatória.
- Adaptar a metodologia proposta para problemas com restrições.
- Criar um modelo no qual as populações das ilhas podem chegar a 0 soluções, ou seja, as ilhas pouco eficientes podem ser eliminadas durante o processo de evolução.

REFERÊNCIAS

[1] HOLLAND, J. H., “Adaptation in Natural and Artificial Systems”, 1975.

[2] FRIEDBERG, R. M., “A learning machine”, 1958.

[3] BREMERMANN, H., “Self-organizing systems”, 1962.

[4] FOGEL, L., A.J., O., M.J., W., “Artificial intelligence through simulated evolution”, 1966.

[5] BEYER, H.-G., SCHWEFEL, H.-P., “Parallel genetic algorithms, population genetics and combinatorial optimization”, 2002.

[6] JONG, K. D., “Analysis of the behavior of a class of genetic adaptive systems”, 1975.

[7] GOLDBERG, D. E., “Genetic algorithms in search, optimization and machine learning”, 1989.

[8] WHITLEY, D., “A genetic algorithm tutorial”, 1994.

[9] GONG, Y.-J., CHEN, W.-N., ZHAN, Z.-H., ZHANG, J., LI, Y., ZHANG, Q., LI, J.-J., “Distributed evolutionary algorithms and their models: A survey of the state-of-the-art”, 2015.

[10] CANTU-PAZ, E., “Master-Slave parallel genetic algorithms”, 2001.

[11] CANTU-PAZ, E., “Designing Efficient Master-Slave Parallel Genetic Algorithms”, 1997.

[12] TANABE, R., FUKUNAGA, A., “Evaluation of a randomized parameter setting strategy for island-model evolutionary algorithms”, 2013.

[13] PIERREVAL, H., “Distributed evolutionary algorithms for simulation optimization”, 2000.

[14] HIDALGO, J. I., DE VEGA, F. F., LANCHARES, J., LOMBRAÑA-GONZÁLEZ, D., “Is the island model fault tolerant?” 2007.

[15] REINHOLD, V. N., “Handbook of genetic algorithms”, 1991.

[16] STORN, R., PRICE, K., "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces", 1997.

[17] KENNEDY, J., "Particle swarm optimization", 2011.

[18] BEYER, H.-G., SCHWEFEL, H.-P., "Evolution strategies – A comprehensive introduction", *Natural Computing*, v. 1, n. 1, pp. 3–52, Mar 2002.

[19] GEEM, Z. W., KIM, J.-H., LOGANATHAN, G. V., "A new heuristic optimization algorithm: Harmony Search", 2001.

[20] GUSTAFSON, S., "The Speciating Island Model: An alternative parallel evolutionary algorithm", 2006.

[21] SKOLICKI, Z., JONG, K. D., "The influence of migration sizes and intervals on island models", 2005.

[22] RUCINSKI, M., "On the impact of the migration topology on the Island Model", 2010.

[23] DUARTE, G., "A dynamic migration policy to the Island Model", 2017.

[24] CHEN, Y.-W., "A parallel genetic algorithm based on the island model for image restoration", 1996.

[25] WHITLEY, D., RANA, S., HECKENDORN, R. B., "The island model genetic algorithm: on separability, population size and convergence", 1999.

[26] JEDRZEJOWICZ, P., "An island based evolutionary algorithm for maximizing schedule reliability", 2000.

[27] SKOLICKI, Z., "Improving evolutionary algorithms with multi-representation island models", 2004.

[28] LI, C., "An island based hybrid evolutionary algorithm for optimization", 2008.

[29] LARDEUX, F., "A dynamic island-based genetic algorithms framework", 2010.

[30] MENG, Q., "Dynamic island model based on spectral clustering in genetic algorithm", 2018.

[31] DUARTE, G. R., "Política de migração para Metaheurísticas híbridas usando Modelo Paralelo de Ilhas", 2019.

[32] BISCANI, F., "A Global Optimisation Toolbox for Massively Parallel Engineering Optimisation", 2010.

[33] LIANG, J. J., QU, B. Y., SUGANTHAN, P. N., "Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization", 2014.

[34] DOLAN, E. D., MORÉ, J. J., "Benchmarking optimization software with performance profiles." *Math. Program.*, v. 91, n. 2, pp. 201–213, 2002.