UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS CRATEÚS

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

IASMIN SANTOS CARVALHO SABA

ANÁLISE DE DESEMPENHO DAS METAHEURÍSTICAS ALGORITMO GENÉTICO E ENXAME DE PARTÍCULAS APLICADAS AO PROBLEMA DE OTIMIZAÇÃO DE PORTFÓLIO FINANCEIRO

CRATEÚS

2024

IASMIN SANTOS CARVALHO SABA

ANÁLISE DE DESEMPENHO DAS METAHEURÍSTICAS ALGORITMO GENÉTICO E ENXAME DE PARTÍCULAS APLICADAS AO PROBLEMA DE OTIMIZAÇÃO DE PORTFÓLIO FINANCEIRO

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

Orientadora: Prof. Ma. Lisieux Marie Marinho Dos Santos Andrade

Coorientador: Prof. Me. Luiz Alberto do Carmo Viana

CRATEÚS

2024

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

S117a Saba, Iasmin Santos Carvalho.
Análise de desempenho das metaheurísticas algoritmo genético e enxame de partículas aplicadas ao problema de otimização de portfólio financeiro / Iasmin Santos Carvalho Saba. – 2024.
46 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Crateús, Curso de Ciência da Computação, Crateús, 2024.
Orientação: Prof. Me. Lisieux Marie Marinho Dos Santos Andrade.
Coorientação: Prof. Me. Luiz Alberto do Carmo Viana.

1. problema de otimização de portfólio. 2. algoritmos genéticos. 3. algoritmos de enxame de partículas. I. Título.

CDD 004

IASMIN SANTOS CARVALHO SABA

ANÁLISE DE DESEMPENHO DAS METAHEURÍSTICAS ALGORITMO GENÉTICO E ENXAME DE PARTÍCULAS APLICADAS AO PROBLEMA DE OTIMIZAÇÃO DE PORTFÓLIO FINANCEIRO

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

BANCA EXAMINADORA

Prof. Ma. Lisieux Marie Marinho Dos Santos Andrade (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Me. Luiz Alberto do Carmo Viana (Coorientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Rennan Ferreira Dantas
Universidade Federal do Ceará (UFC)

Prof. Me. Marciel Barros Pereira
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

À Deus, que me deu forças durante essa jornada e me fez continuar.

À minha familia, especialmente minha avó Diomarina que foi a razão pela qual decidi seguir esse caminho.

À Profa. Ma. Lisieux Marie Marinho dos Santos Andrade, por ter aceito me orientar, ter tido paciência e ser o lado forte de quando precisei. Te chamar pra ser minha orientadora foi a melhor escolha que eu poderia ter feito.

Ao Prof. Me. Luiz Alberto do Carmo Viana, por ter aceito me coorientar e ser o apoio nos momentos que duvidei de mim. Vocês foram o time perfeito que escolhi desde o primeiro dia.

À minha amiga Ana Luiza, pela parceria durante essa jornada e por ter ficado ao meu lado durante meus longos períodos de tristeza e reclamação.

À todos os professores que fizeram parte da minha jornada acadêmica.

À todos meus amigos e colegas que estiveram comigo durante a graduação, pelo companheirismo e por tornar essa experiência mais leve e especial.

RESUMO

O Problema de Otimização de Portfólio Financeiro envolve determinar a proporção ideal e a melhor alocação de recursos em ativos financeiros, objetivando maximizar os lucros e minimizar os riscos. Ao longo dos anos, o modelo clássico para esse problema, introduzido por Markowitz (1952) foi aprimorado com diferentes adições de restrições que buscaram torná-lo mais alinhado à realidade do investidor, como, por exemplo, as restrições de cardinalidade da carteira, de piso e teto do investimento para cada ativo, e de orçamento. Dessa forma, a complexidade do problema aumenta quando trata-se de encontrar soluções exatas, o que sugere o uso de procedimentos metaheurísticos, que se destacam por sua capacidade para encontrar boas soluções com a flexibilidade de se adaptar para diferentes modelos do problema. Nesse contexto, o presente trabalho investiga o desempenho e compara duas meteheurísticas quando aplicadas ao Problema de Otimização de Portfólio Financeiro, o Enxame de Partículas (PSO) e o Algoritmo Genético (GA), sendo último aperfeiçoado com um procedimento de reparação inspirado na literatura. Para a execução dos experimentos, foram utilizadas instâncias clássicas e as métricas *fronteira eficiente*, *média da distância euclidiana*, *erro médio de retorno*, *erro médio da variância* e *tempo de execução*. Após 75 casos de análises, verificou-se que o Algoritmo Genético demonstrou um melhor desempenho.

Palavras-chave: problema de otimização de portfólio; algoritmos genéticos; algoritmos de enxame de partículas.

ABSTRACT

The Financial Portfolio Optimization Problem involves determining the optimal proportion and the best allocation of resources in financial assets, aiming to maximize profits and minimize risks. Over the years, the classical model for this problem, introduced by Markowitz (1952), has been improved with different additions of restrictions that sought to make it more aligned with the investor's reality, such as, for example, the cardinality restrictions of the portfolio, the floor and ceiling of the investment for each asset and the budget. Thus, the complexity of the problem increases when it comes to finding exact solutions, which suggests the use of metaheuristic procedures, which stand out for their ability to find approximate and flexible solutions to adapt to different models of the problem. In this context, the present work investigates the performance and compares two metaheuristics when applied to the Portfolio Optimization Problem, the Particle Swarm (PSO) and the Genetic Algorithm (GA), the latter being improved with a component procedure inspired by the literature. To perform the experiments, classical instances and the metrics of efficient frontier, mean Euclidean distance, mean return error, mean variance error and execution time were used. After 75 cases of analysis, it was found that the Genetic Algorithm demonstrated better performance.

Keywords: portfolio optimization problem; genetic algorithms; particle swarm algorithms.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Investimentos financeiros são inerentes ao mundo capitalista, nos quais indivíduos, empresas ou países realizam aquisições visando obter lucro ao longo do tempo. Esses investimentos tornam-se possíveis através de ativos financeiros, isto é, recursos com valor econômico como ações, títulos, moedas, e derivativos, que trazem a expectativa de proporcionar um benefício futuro. A combinação desses ativos financeiros é chamada de Carteira de Investimentos ou Portfólio Financeiro (Mansini R. e Speranza (2015)). Dessa forma, por ter uma característica combinatória, na área da Otimização Computacional, este é definido como um problema, denominado Otimização de Portfólio.

O Problema de Otimização de Portfólio Financeiro tem por objetivo determinar a melhor forma de alocar ativos financeiros na carteira de investimento, decidindo quais são os escolhidos e a proporção de capital aplicado em cada um de acordo com um valor disponível, buscando, dessa forma, maximizar o lucro e diminuir o risco. O lucro é o retorno que o investidor espera e se caracteriza por ser o saldo positivo obtido após descontar o investimento inicial. O risco, por sua vez, representa a incerteza desse investimento, e portanto, o equilíbrio entre esses dois parâmetros deve ser avaliado, buscando evitar um lucro menor do que o esperado ou até mesmo negativo, resultando em prejuízo (Mansini R. e Speranza (2015)).

Outro conceito importante e que busca diminuir o risco do investidor, é o de diversificação. Para compreendê-lo é preciso entender que alocar todo o orçamento disponível em um único ativo pode ser uma escolha de alto risco, assim como investir em um conjunto de ativos que são influenciáveis por fatores econômicos parecidos. Isso acontece pois, em geral, há uma correlação não nula entre o desempenho de pares de ativos (Mansini R. e Speranza (2015)). Observa-se, então, que é importante escolher diferentes tipos de ativos para compor a carteira, pois essa diversificação garante um grau de certeza maior, portanto, um menor risco.

Nesse contexto, um dos primeiros estudos na área de Otimização de Portfólio Financeiro foi feito por Markowitz (1952) que revolucionou a area de estudo e introduziu o conceito de diversificação discutido acima, incentivando assim, outros trabalhos na área e mudando a forma de visualizar o problema. Markowitz (1952) propôs um modelo denominado média-variância (MV) com o objetivo de maximizar o retorno, medido pela média estatística de valor esperado, e minimizar o risco da carteira de investimento, mensurado através da variância do retorno (Sharpe (1964)). Segundo alguns estudiosos, a variância utilizada por Markowitz (1952) não seria ideal, pois levava em consideração os valores acima e abaixo da média.

Dessa forma, surgiram outros modelos utilizando diferentes formas de dimensionar o risco. Tendo como base os desvios negativos, algumas dessas métricas são: Valor em Risco Condicional (*Conditional Value-at-Risk - CVaR*) que corresponde a medida de risco estimada pela média ponderada das perdas desde o pior resultado até o ponto de corte do valor em risco (*Value-at-Risk-VaR*), tendo como principal exemplo o modelo proposto por Rockafellar *et al.* (2000), e Roman *et al.* (2007), entre outros; Momento Parcial Inferior (Lower Partial Moment - LPM), que está relacionada a distribuição de probabilidade dos retornos, presente em trabalhos como Estrada (2007); Desvio Médio Absoluto (Mean Absolute Deviation - MAD) presente em trabalhos de Konno e Yamazaki (1991) e Speranza (1996).

Buscando tornar o problema mais próximo da realidade do investidor, modelos com novas restrições foram abordados, como cardinalidade da carteira, limites inferiores e superiores de investimento para cada ativo, de lote mínimo de transação, pré-seleção de ativos, entre outras. Porém, é importante observar que a presença de restrições adicionais pode tornar o problema computacionalmente mais complexo (Chang *et al.* (2000)).

Considerando o cenário das pesquisas na área, essa adição de restrições revela que métodos exatos tornam-se difíceis de serem utilizados. Desse modo, procedimentos heurísticos e metaheurísticos são boas estratégias a serem utilizadas, dada suas características já conhecidas, por serem executados em tempo polinomial razoável e conseguirem encontrar boas soluções (Chang *et al.* (2000), Crama e Schyns (2003), Schaerf (2002)). Além disso, apesar da evolução de técnicas como Machine Learning (ML), Inteligência Artificial (IA) e Redes Neurais que trouxeram ganhos significativos para o Problema de Otimização de Portfólio, sendo utilizados para revolver modelos complexos, metaheurísticas ainda são utilizados em conjunto com essas técnicas (Thakkar e Chaudhari (2021), Kim e Shin (2007), Versace *et al.* (2004)).

Nesta perspectiva, a presente pesquisa apresenta os resultados obtidos com o desenvolvimento, aplicação e comparação de desempenho de duas metaheurísticas baseadas em abordagens clássicas: o Enxame de Partículas e o Algoritmo Genético(do inglês *Particle Swarm Optimization* - PSO e *Genetic Algorithm* - GA), respectivamente. O último é enriquecido com um procedimento de reparação inspirado no trabalho de Jalota e Thakur (2018).

1.1 Objetivo Geral

Avaliar e comparar os desempenhos das metaheurísticas Enxame de Partículas (PSO) e Algoritmo Genético (GA) na resolução do Problema de Portifólio Financeiro, por meio das

métricas *fronteira eficiente*, *média da distância euclidiana*, *erro médio de retorno*, *erro médio da variância* e *tempo de execução*.

1.1.1 Objetivos Específicos

- Analisar o comportamento individual das metaheurísticas PSO e GA com base na *fronteira eficiente*;
- Avaliar os desempenhos dos algoritmos PSO e GA em relação às métricas: *média da distância euclidiana*, *erro médio de retorno*, *erro médio da variância* e *tempo de execução* dos experimentos;
- Comparar os melhores desempenhos das metaheurísticas PSO e GA considerando a *fronteira eficiente*.

1.2 Estrutura do Trabalho

O presente trabalho é composto por sete capítulos:

- Capítulo 1 - Introdução: Apresenta o contexto da pesquisa e os objetivos a serem alcançados.
- Capítulo 2 - Fundamentação Teórica: Trata os elementos essenciais para a compreensão da pesquisa.
- Capítulo 3 - Problema de Otimização de Portfólio: Este capítulo discute a definição do problema de Portfólio Financeiro.
- Capítulo 4 - Trabalhos Relacionados: Apresenta dois trabalhos que inspiraram a aplicação de procedimentos, variáveis relevantes e as métricas utilizadas na fase de análise da pesquisa.
- Capítulo 5 - Abordagens Utilizadas: Neste capítulo, são descritas as metaheurísticas Genetic Algorithm (GA) e Particle Swarm Optimization (PSO), implementadas para a resolução do problema.
- Capítulo 6 - Experimentos e Resultados: Apresenta os resultados e discussões dos 75 casos de testes realizados nesta pesquisa, considerando as instâncias, o cenário dos experimentos e as métricas utilizadas.
- Capítulo 7 - Conclusão: Apresenta a síntese e as contribuições da pesquisa e sugere alguns trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

O termo Otimização Computacional (OC) refere-se à utilização de técnicas matemáticas com o objetivo de encontrar a melhor solução dado um conjunto de candidatos que por vezes podem ser condicionados à restrições. Conforme Gamarra e Guerrero (2015), a otimização procura encontrar os melhores valores disponíveis que satisfaçam uma função objetivo de acordo com um domínio definido ou um conjunto de restrições. Desta forma, para melhor compreensão do presente trabalho, este capítulo irá se aprofundar no conceito de Problemas de Otimização, além de apresentar Soluções e Abordagens para Problemas de Otimização Computacional.

2.1 Problemas de Otimização Computacional

Os problemas de otimização se dividem em algumas categorias. Há os de domínio contínuo, chamados de problemas de otimização contínua, em que geralmente se procura por um conjunto de números reais ou uma função. Existem outros com domínio discreto, chamados de problemas de otimização combinatória ou discreta, nos quais se busca por um objeto dentro de um conjunto finito ou, possivelmente, um conjunto infinito contável. Ainda no contexto dos domínios, há os que contemplam a combinação dos domínios contínuos e discretos. Os problemas de otimização também podem ser categorizados pelo seu objetivo: quando único, chamado de monoobjetivo; ou quando são dois ou mais, chamado de multiobjetivo.

A formalização de um problema de otimização se dá por meio de um modelo, denominado *modelo padrão* ou *forma padrão*:

$$\textit{Minimizar} f(x) \tag{2.1}$$

Sujeito às restrições:

$$g_j(x) \leq 0, \quad j = 1,2,\ldots,m \tag{2.2}$$

$$h_j(x) = 0, \quad j = 1,2,\ldots,r \tag{2.3}$$

Em que $f(x)$, $g(x)$ e $h(x)$ são funções escalares do vetor real $x = [x_1, x_2, ..., x_n] \in \mathbb{R}$ e os seus elementos chamamos de variáveis. $f(x)$ é a função objetivo, $g_j(x)$ denota as restrições de desigualdade e $h_j(x)$, as de igualdade. Desta maneira, os pontos que respeitam todas as restrições são chamados de pontos viáveis.

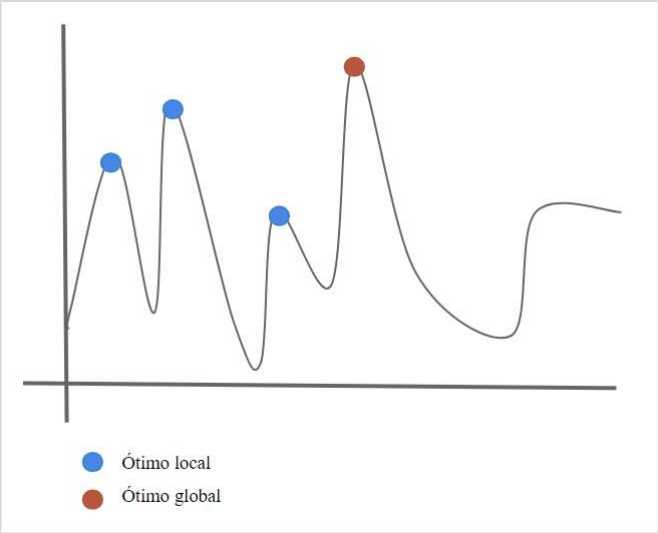
A definição de uma instância de uma problema de otimização é dada por um par (X,c), em que X é um domínio de pontos viáveis; c é a função de custo, um mapeamento $c : X \to \mathbb{R}^1$, onde $\mathbb{R}^1$ significa que é um conjunto de pontos escalares. Desta forma, para um problema de minimização busca-se encontrar uma instância em que $x \in X$ tal que $c(x) \leq c(y)$ e para um problema de maximização isso ocorre nos momentos em que $c(x) \geq c(y)$, para todo $y \in X$ (Papadimitriou e Steiglitz (1998)).

2.2 Soluções

Dado um ponto viável $x \in X$ de um determinado problema, chamamos de vizinhança de x os pontos que estão próximos e para isso uma função $N : X \to 2^X$ relaciona todo $x \in X$ com seu conjunto de vizinhos $N(x) \subseteq X$, ou seja $N(x)$ é a vizinhança de x. Segundo Papadimitriou e Steiglitz (1998), em muitos problemas de otimização combinatória, a escolha de N pode depender da estrutura de X. Desse modo, dada uma instância de um problema de otimização e uma vizinhança N, uma solução viável é dita um ótimo local se para problemas de minimização $c(x) \leq c(g),\ \forall g \in N(x)$ e para de maximização $c(x) \geq c(g),\ \forall g \in N(x)$.

Na Figura 1 os ótimos locais estão representados com a cor azul, considerando um problema de maximização e pode-se perceber que na vizinhança desses pontos eles são os melhores. Chama-se x de ótimo local estrito se $c(x) < c(g),\ \forall g \in N(x)$ ou $c(x) > c(g),\ \forall g \in N(x)$ para problemas de minimização e maximização, respectivamente. Se o espaço de busca possuir vários ótimos locais, o menor deles (ou maior, caso o problema seja de maximização) será o ótimo global, isto é, o melhor valor que satisfaça a função objetivo, desde que essa não seja ilimitada. Na Figura 1 está sendo considerado um problema de maximização e o ótimo global é o ponto mais alto representado pela cor vermelho, logo não existe outro ponto no gráfico maior, ou seja, que possua uma função de custo maior que a desse ponto.

Figura 1 – Ótimo global e ótimos locais, dado uma função de maximização



Fonte: Próprio autor

Tendo os elementos para a formalização de soluções de um Problema de Otimização apresentados, na próxima seção serão abordadas algumas técnicas empregadas para o encontro de tais soluções.

2.3 Abordagens para Problemas de Otimização Computacional

Algoritmos que resolvem problemas de otimização computacional possuem diferentes abordagens para chegar em uma solução, seja procurando como resultado a solução ótima ou alguma suficientemente próxima. Nesta seção serão abordados os conceitos de Abordagem Exata, Heurísticas e Metaheurísticas.

2.3.1 Abordagem Exata

Na abordagem exata, os algoritmos buscam, dentro das restrições definidas, a melhor configuração possível para determinado problema, isto é, buscam o ótimo global. Diferentes métodos com diferentes técnicas, utilizam essa abordagem, como: Programação Inteira, Linear e Dinâmica. A seguir, será apresentado um pouco sobre cada uma delas.

Programação Dinâmica consiste em dividir o problema principal em subproblemas de mesma natureza, de tal forma que as soluções desses subproblemas compõem a solução do problema maior. Para problemas de menor escala e que possuem um número pequeno de subconjuntos, com uso deste procedimento pode-se alcançar maior eficiência devido ao menor

consumo de memória e ao tempo computacional (Cormen *et al.* (2009)). Algoritmos podem ser desenvolvidos usando programação dinâmica para diversos problemas, a exemplo do segmento de soma máxima e o do caixeiro viajante com métrica triangular (Woeginger (2003)).

A Programação Linear modela um problema através de expressões obrigatoriamente lineares. A popularidade desse tipo de problema decorre da fase de formulação da análise, pois uma boa quantidade de restrições e objetivos são naturalmente lineares, além disso, essa formulação é relativamente fácil de definir quando comparada a outros tipos de modelagem. Para definir o funcionamento da programação linear de forma mais simples, um mecanismo é a utilização da forma geométrica para visualizar-se a relação desse tipo de problema com poliedros convexos. Poliedro é descrito como o conjunto $x : Ax \leq b$ em que x e b são vetores e A é uma matriz ou geometricamente, significa uma forma multidimensional com faces de diferentes dimensões. Por ser convexo, significa que para cada dois pontos pertencentes desse poliedro, um segmento de reta entre eles também pertence ao mesmo. Quando o conjunto de soluções viáveis é um poliedro fechado, uma das soluções ótimas é necessariamente um vértice (vetor que está na extremidade), e portanto satisfaz a função objetivo e todas as restrições. Um procedimento bastante eficiente utilizado para resolver tais problemas é o Método Simplex, que basicamente caminha pelos vértices do poliedro, buscando a solução ótima para o problema (Luenberger *et al.* (1984)).

A Programação Inteira é um ramo da Programação Linear que modela problemas lineares e suas variáveis são obrigatoriamente inteiras, o que em geral torna-os muito mais difíceis de se resolver dado o elevado custo computacional. Um método muito conhecido é o de *Branch and Bound*, no qual há um particionamento do espaço de soluções original, onde cada subespaço corresponde a um subproblema derivado do problema inicial. Cada subespaço é criado adicionando novas restrições que criam limites superiores e inferiores. Esse passo, geralmente envolve resolver a versão relaxada (versão mais simples, podendo ter menos restrições) do problema e os limites ignoram a restrição de integralidade das variáveis. Ao resolver ou eliminar todos os subproblemas, a solução ótima é encontrada (Wolsey e Nemhauser (2014)).

2.3.2 Abordagem Heurística

Utilizar métodos exatos pode não ser a melhor forma de resolver determinados problemas, pois dada sua natureza combinatória, no pior caso, todas as soluções precisam ser analisadas. Para conjuntos maiores, o cenário se torna mais desafiador, podendo demandar muito

tempo e um alto custo computacional. Porém, esses problemas podem ser resolvidos em um tempo razoável e menor custo computacional utilizando métodos heurísticos, onde a solução encontrada pode não necessariamente ser a ótima, mas uma suficientemente próxima da ótima e boa para o problema em questão. Como a qualidade da solução está inerente ao seu custo e ao problema, muitas vezes torna-se mais interessante a utilização de uma abordagem heurística.

Os procedimentos heurísticos podem ser divididos em construtivos e de busca local. As heurísticas construtivas, de acordo com Silver *et al.* (1980), têm por característica construir a solução elemento por elemento a partir de uma solução vazia. A forma que esses elementos são escolhidos e adicionados depende do problema considerado. Além disso, Blum e Roli (2008) afirmam que esses são métodos rápidos e que finalizam sua atividade quando a solução está completa (ocorre quando encontramos todos os valores do conjunto solução) ou um critério de parada é satisfeito. Uma conhecida classe de heurísticas construtivas são os algoritmos gulosos que escolhem o "melhor" elemento para ser adicionado à solução, sendo a definição de *"melhor"* ou *"pior"* dada de acordo com a função objetivo.

Os métodos de busca local ou refinamento caracterizam-se pela busca no espaço de soluções em que, a partir de uma solução inicial que pode ser obtida aleatoriamente ou através de uma heurística construtiva, procura-se em sua vizinhança outras soluções promissoras. Uma solução vizinha desse valor inicial, por exemplo, pode ser aquela obtida após alguma modificação na solução local, por isso este procedimento é conhecido como *busca local*. Ou seja, a cada iteração a solução é refinada, buscando a melhor solução. De acordo com Hertz e Widmer (2003) esse processo pode ser visto como um caminho em um grafo direcionado $G = (S,A)$, onde S é o conjunto de soluções e haverá um arco (s,s') em A que é o conjunto de arestas se, e somente se, s' estiver na vizinhança de s. Além disso, é importante ressaltar que é preciso determinar a definição de vizinhança que será adotado no problema em questão.

Os procedimentos heurísticas embora eficientes estão relacionados à problemas específicos, logo, muitas vezes podem ser particulares e acabam se tornando ineficientes se aplicados a uma classe mais ampla de problemas. Com o avanço das pesquisas, surgiram as metaheurísticas que apresentam abordagem mais generalista, como será explicada na seção a seguir.

2.3.3 Metaheuristicas

O conceito de Metaheuristica surgiu nos anos 1970, combinando métodos heurísticos básicos em frameworks de alto nível (Blum e Roli (2008)). Assim como as heurísticas, as metaheuristicas não encontram necessariamente a solução ótima, mas sim alguma suficientemente boa em um custo computacional razoável. Para a sua elaboração, normalmente, são fáceis de serem implementadas em comparação a outras técnicas clássicas (Blum e Roli (2008)). Desta forma, podemos caracterizar metaheuristicas como estratégias que orientam o processo de busca e que seu objetivo é explorar de forma eficiente o espaço de busca para encontrar soluções. As heurísticas e metaheuristicas possuem objetivos semelhantes, no entanto, as metaheuristicas apresentam abordagem generalista, não precisando conhecer detalhes do problema para iniciar a busca por uma solução, isto é, são facilmente adaptáveis.

Quanto à sua aplicação, alguns problemas de otimização que podem ser resolvidos com algoritmos exatos tornam-se complexas com restrições adicionais, tornando complexo ou até mesmo intratável a tarefa de continuar utilizando essa mesma abordagem, como é o caso do problema trabalhado nesta pesquisa e que será apresentado no Capítulo 3. Para estes casos, as metaheuristicas ganharam espaço, tornando mais fácil encontrar boas soluções. Alguns procedimentos conhecidos por seus desempenhos são: Busca Local Iterada, *Simulated Annealling*, Busca Tabu e os algoritmos de Computação Natural, sendo esse último um campo de estudo reservado aos algoritmos que são inspirados na natureza. A seguir, serão apresentados alguns procedimentos que são utilizados nesta pesquisa.

2.3.3.1 Algoritmo Genético - GA

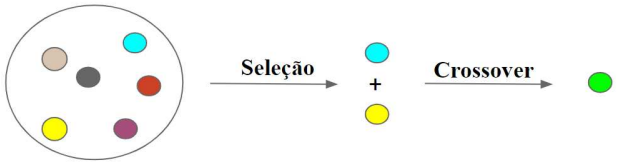
Algoritmos evolutivos baseiam-se na teoria da evolução em que considera que os seres vivos evoluíram de espécies simples Darwin *et al.* (1859). De acordo com o princípio da seleção natural proposta pela teoria, as espécies que sobrevivem e produzem mais descendentes são aquelas que estão mais adaptadas com o ambiente. Essa família de algoritmos, incorporam diferentes abordagens, tais como Algoritmos Genéticos (do inglês, *Genetic Algorithm* - GA), Programação Evolutiva, Estratégias Evolutivas e Programação Genética (Reeves e Rowe (2002), Katoch *et al.* (2021)).

Dentre os procedimentos citados, o GA é o mais conhecido e amplamente utilizado para problema abordado nesse trabalho. Criado em 1975 por John Holland, o procedimento

inicia seu funcionamento com uma população que normalmente é criada randomicamente Reeves e Rowe (2002). Alguns indivíduos dessa população serão selecionados para serem os pais da próxima geração, porém, para serem escolhidos, é necessário uma forma de avaliação ou critério de escolha. Para isso, são atribuídos a eles um valor de adaptação obtido por meio da função *fitness*. Com base nessa função, os pais são selecionados e é feita a reprodução genética, ou de maneira simples, uma “mistura” dos valores dos pais; esse mecanismo é denominado *crossover*.

Na Figura 2 é considerado como objetivo do problema encontrar um conjunto de cores variadas. O exemplo seleciona dois indivíduos da geração anterior e faz o *crossover*, ou seja, a "mistura"das cores para obter uma nova, no caso, ciano e amarelo resulta na cor verde.

Figura 2 – Exemplo de crossover



Fonte: Próprio autor

Os indivíduos também podem sofrer mutações durante o processo reprodutivo, características que não são herdadas dos pais são adquiridas de outras formas. Por exemplo, um algoritmo que soma um valor x em alguns indivíduos. O processo de mutação é importante pois ajuda o algoritmo a escapar de ótimos locais. Segundo Linden (2008) auxilia na obtenção de uma maior variedade genética e assim como na natureza é aplicado com uma frequência menor do que o crossover. No exemplo da Figura 3, ainda considerando o mesmo problema da Figura 2, o processo de mutação é caracterizado como o acréscimo de um tom no indivíduo, tornando-o mais escuro, ou seja, não está relacionado aos pais, dessa forma, temos um verde escuro.

Figura 3 – Exemplo de mutação

Fonte: Próprio autor

Aliados a operações de *crossover* e mutação, algumas técnicas são utilizadas. Uma das mais populares, é o *elitismo* que guarda os melhores da geração anterior para a próxima, buscando evitar a perca de informações importantes presentes nesses indivíduos e que podem, por

ventura, serem perdidas durante os processos anteriores (Bento e Kagan (2008)). Outra técnica é a *alocação de fitness relativa*, que visa promover a diversidade por meio do compartilhamento do valor de fitness entre os indivíduos, dessa forma, tende a evitar que soluções similares dominem o processo evolutivo Sareni e Krahenbuhl (1998).

Para o aprimoramento da operação de *seleção*, uma técnica muito utilizada é a de *ranking*, que tem como objetivo evitar que indivíduos com valores *fitness* excepcionalmente altos dominem o processo evolutivo. Para isso, ao invés de utilizar diretamente a *fitness*, é feita uma classificação relacionando esses valores com posições em um rank. Dessa forma, como a diferença se torna menor quando considera as posições, os indivíduos com valores excepcionais diminuem as chances de dominarem sempre. Além disso, a implementação desse rank pode ser feita de diversas formas, usando, por exemplo, funções lineares ou exponenciais.

Após os passos de seleção, reprodução e variação genética, é obtido uma geração composta por novos indivíduos que são mais adaptados do que a geração anterior. Dessa forma, fazendo uso dessas operações e técnicas citadas anteriormente, o algoritmo genético itera conforme uma condição estabelecida (geralmente um número máximo de iterações), até encontrar a melhor solução possível. No pseudocódigo 1 é apresentado esse modelo básico do GA.

Algoritmo 1 : GA

```
1: Inicializar a população
2: enquanto Condição não for satisfeita faça
3:     Avaliar fitness
4:     Selecionar os cromossomos que irão passar pelos operadores
5:     Aplicar crossover
6:     Aplicar mutação
7:     Atualizar população
8: fim enquanto
9: retorna Melhor indivíduo encontrado
```

Fonte: Próprio autor.

2.3.3.2 Algoritmos de Enxame de Partículas - PSO

Na década de 90, cientistas de áreas da computação e zoologia estudavam o comportamento psicológico e social de determinadas espécies. Eles perceberam as interações que bandos de aves e cardumes de peixes realizam nas suas atividades cotidianas e que possuem uma forma de consciência coletiva, onde cada integrante aprende com os outros do grupo a

melhor direção a ser seguida. O algoritmo de Otimização de Enxame de Partículas ou do inglês *Particle Swarm Optimization* (PSO) é uma metaheurística que simula o comportamento social, inclusive da sociedade humana, buscando modelar a exploração de um espaço relacionado a algum problema por uma população de indivíduos (Kennedy (1997))

No PSO cada indivíduo é tratado como partícula sem volume em um espaço de busca D-dimensional (Shi e Eberhart (1999)) e funciona da seguinte forma: cada partícula é caracterizada por sua posição e um vetor de mudança de posição, chamada velocidade. Dada uma população inicial, o algoritmo atualiza, a cada iteração, a velocidade de cada partícula, baseando-se na velocidade das outras. O comportamento de uma partícula depende do comportamento de seus vizinhos. Portanto, para calcular essa posição e velocidade definidas como p e v, respectivamente, utiliza-se a Equação (2.4). Em que para um conjunto P de partículas, tem-se a posição de k na iteração i ($p_{k,i}$), sabendo que $k \in P$.

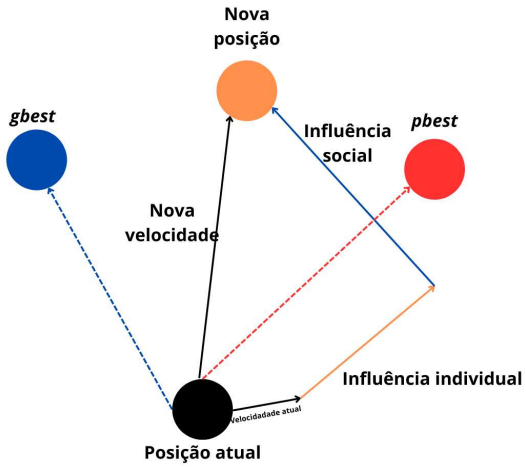
$$v_{k,i+1} = v_{k,i} + c_1 * rand() * (pbest_k - p_{k,i}) + c_2 * rand() * (gbest - p_{k,i}) \tag{2.4}$$

$$p_{k,i+1} = p_{k,i} + v_{k,i+1} \tag{2.5}$$

Pontua-se que, $pbest_k$ é a melhor posição explorada pela partícula, $gbest$ é a melhor posição explorada entre todas as partículas, os valores de $rand()$ correspondem uma função aleatória dentro do intervalo $[0,1]$ e que, c_1 e c_2 são duas constantes de aceleração referentes ao melhor individual e global, respectivamente.

Dessa forma, é possível verificar a ilustração para o comportamento do PSO na Figura 4, que representa a influência do *gBest* ao determinar a nova posição de uma partícula k. A circunferência de cor preta representa a partícula k que terá a posição modificada; a de cor azul representa o *gBest*, e a de cor vermelha representa o *pBest*. A direção que a partícula tende a seguir, sem a influência do *gBest*, é representada pela seta laranja. Finalmente, a circunferência laranja indica a nova posição da partícula considerando a influência global.

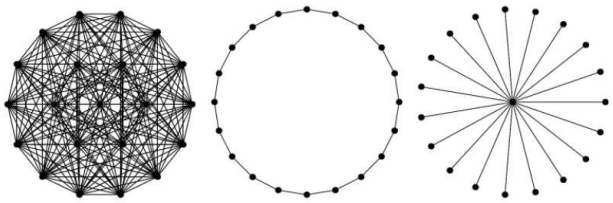
Figura 4 – PSO



Fonte: Próprio Autor

Um elemento que compõe o PSO e possui relevância é a topologia. Topologia corresponde a um subconjunto de partículas com as quais uma partícula irá trocar informações. Como no PSO o comportamento de uma partícula depende das outras partículas, percebe-se que a escolha da topologia influência na performance do algoritmo. A versão básica do PSO utiliza uma topologia global (ou totalmente conectada) como estrutura, permitindo que todas as partículas se comuniquem entre si, o que nem sempre é interessante. Observa-se que, se existir uma função com muitos máximos e/ou mínimos, uma partícula pode ficar "presa" em algum deles e influenciar que outras partículas também sigam o seu comportamento, fazendo o algoritmo ter uma convergência prematura. Por outro lado, se uma topologia local permitir que as partículas apenas compartilhem informações com um subconjunto é possível evitar que essa convergência prematura aconteça, não anulando a possibilidade que ocorra, porém de forma mais lenta. A Figura 5 apresenta alguns exemplos de topologias. Outro elemento neste aspecto da topologia é o número de vizinhos que uma partícula possui, este fator caracteriza o grau dela e o número médio de partículas vizinhas corresponde ao grau médio da topologia, representando o grau de socialização do enxame, como definido em Yin *et al.* (2015).

Figura 5 – Topologia totalmente conectada; topologia em anel; topologia em estrela



Fonte: Yin *et al.* (2015)

O pseudocódigo 2 ilustra o funcionamento do *PSO*, utilizando os conceitos e fórmulas discutidos anteriormente. Na linha 1, ocorre a inicialização, momento em que a população é criada e as partículas recebem posições aleatórias. Os valores de *pbest* são definidos por essas posições iniciais, e o melhor valor entre elas é o *gbest*. Da linha 2 à linha 16, o laço de repetição executa as atualizações de *pbest*, *gbest*, e da velocidade, utilizando o cálculo 2.4. Ao final do processo a posição é atualizada com base na fórmula 2.5.

Algoritmo 2 : PSO

```
1:  Inicialize a população de partículas
2:  enquanto condição não for satisfeita faça
3:      Calcular fitness
4:      para cada partícula faça
5:          se o valor for melhor que pbest então
6:              Atualizar pbest
7:          fim se
8:          se o valor for melhor que gbest então
9:              Atualizar gbest
10:         fim se
11:     fim para
12:     para cada partícula faça
13:         Atualizar velocidade
14:         Atualizar posição
15:     fim para
16: fim enquanto
17: retorna gBest como a melhor solução encontrada
```

Fonte: Próprio autor.

Conhecido os conceitos dos principais elementos que fundamentam a presente pesquisa, no próximo capítulo será apresentada a definição do problema abordado neste trabalho.

3 PROBLEMA DE OTIMIZAÇÃO DE PORTFÓLIO

A forma de analisar o problema de otimização de portfólio teve uma grande mudança com Harry Markowitz que introduziu a chamada teoria do portfólio através do seu trabalho intitulado Portfolio Selection em 1952 (Markowitz (1952)) lhe rendendo o prêmio Nobel da economia em 1990. Markowitz traz a ideia do investidor como um ser racional com o intuito de maximizar os lucros, possuindo aversão ao risco. Ele propôs um modelo denominado de média variância (MV) que pode ser descrito como uma função quadrática que minimiza o risco da carteira. Após seus trabalhos o assunto é visto como um problema de otimização em que há dois objetivos: o retorno esperado que deve ser maximizado; e o risco da carteira que deve ser minimizado (Sharpe (1964), Markowitz (1952)).

A seguir apresenta-se o modelo MV proposto por Markowitz (1952).

$$f(x) = \max \sum_{i=1}^{n} l_i x_i \tag{3.1}$$

$$g(x) = \min \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j \sigma_{ij} \tag{3.2}$$

$$\text{s.t:} \sum_{i=1}^{n} x_i = 1 \tag{3.3}$$

$$x_i \geq 0 \tag{3.4}$$

Nesse modelo, n é o número de ativos que estão sendo considerados, x_i é o percentual de capital investido em um ativo i, σ_{ij} é a covariância entre os ativos i e j, l_i é o retorno médio do ativo i. A função objetivo (3.1) representa o retorno da carteira e deve ser maximizado, já a (3.2) é o risco da certeira utilizando variância e deve ser minimizada. As restrições (3.3) e (3.4) representam que o investimento nos ativos não deve ultrapassar 100% do montante total tanto na carteira, quanto individualmente.

3.1 Otimização de Portfólio com Restrições

A abordagem proposta por Markowitz, apesar de mudar a forma de ver o problema se baseia em um mercado simplista, isto é, não é algo realista, pois na prática outros critérios devem ser consideradas, como alocação de recursos, cardinalidade e custos de transação. No entanto, essas restrições, podem tornar o problema mais complexo. Para o modelo clássico, a adição de restrições de cardinalidade eleva sua complexidade, tornando-se NP-Difícil, como

provado por Moral-Escudero *et al.* (2006) sendo um problema quadrático inteiro misto. Além disso, para métodos exatos podem se tornar inviáveis como observado por Mishra (2012).

Chang *et al.* (2000) adiciona restrições de cardinalidade, orçamento e de limite inferior e superior para a proporção de investimento de cada ativo. Chang *et al.* (2000) mostra que essas inclusões tornam o problema mais complexo devido à descontinuidade da fronteira eficiente causada pelas restrições. O seu modelo é apresentado a seguir.

$$\min \ \lambda \left(\sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \sigma_{ij} \right) - (1-\lambda) \left(\sum_{i=1}^{n} w_i \mu_i \right) \tag{3.5}$$

$$\text{s.t:} \quad \sum_{i=1}^{n} w_i = 1, \tag{3.6}$$

$$\sum_{i=1}^{n} z_i = K, \tag{3.7}$$

$$\varepsilon_i z_i \leq w_i \leq \delta_i z_i, \ \ i = 1, ...N, \tag{3.8}$$

$$z \in [0,1], \ \ i = 1, ...N. \tag{3.9}$$

Onde, K é o número de ativos desejados no portfólio e N é o total de ativos. Para $i = 1, ..., N$ temos que w_i é a proporção de dinheiro investido no ativo i, ε_i e δ_i representam respectivamente a proporção mínima e máxima de dinheiro investido em i e z_i é uma variável de decisão, se $z = 1$ o ativo pertence ao portfólio, caso seja $z_i = 0$ ele não pertence.

A restrição (3.6) representa a restrição orçamentária também presente no modelo MV (3.3). A nova restrição (3.7) é a de cardinalidade e significa que há um número desejados de ativos presentes no portfólio. Já a restrição (3.8) representa o piso e teto de proporção de investimento de um ativo. As duas últimas restrições tornam o modelo mais realista, já que representam questionamentos reais quando investidores montam uma carteira de investimento.

Problemas de otimização de portfólio, utilizando o modelo de restrições discutido neste capítulo, podem ser abordados por meio de diversas metaheurísticas. No próximo capítulo, exploraremos trabalhos que aplicam esse modelo, servindo como base para o desenvolvimento dos métodos propostos nesta pesquisa.

4 TRABALHOS RELACIONADOS

Após o trabalho de Markowitz, o problema de Otimização de Portfólio foi tratado por meio de diversas abordagens. Alterações em seu modelo, além da aplicação de diferentes soluções técnicas, impulsionaram os estudos relacionados ao problema. Na literatura, GA e PSO evidenciaram performances exitosas, como em trabalhos de Jalota (2016) e Yin *et al.* (2015). Esses autores propuseram e estudaram algoritmos modificados, buscando uma maior qualidade das soluções, como apresentado nas seções a seguir.

Ambos os trabalhos utilizam o mesmo modelo (3.5 - 3.9) e bases de dados clássicas, aspectos fundamentais para esta pesquisa. O estudo de Jalota (2016) oferece uma perspectiva mais robusta sobre o uso de algoritmos genéticos, contribuindo diretamente para o desenvolvimento do GA implementado. Por outro lado, Yin *et al.* (2015) apresenta uma análise detalhada dos resultados, que serve de base para a abordagem adotada neste trabalho.

4.1 Genetic algorithm designed for solving portfolio optimization problems subjected to cardinality constraint

Jalota e Thakur (2018) em seu trabalho propõem um algoritmo genético para resolver o modelo proposto por Chang *et al.* (2000) adicionando uma nova restrição de *no short selling* (sem venda a descoberto) que significa que o investidor só pode vender ativos que ele possui. Essa restrição é incluída no modelo como $w_i \geq 0$. Porém, como os autores utilizam os valores de 0.1 e 1 para o limites inferiores e superiores, concluí-se que na prática, o modelo se iguala ao proposto por Chang *et al.* (2000). No algoritmo genético proposto, os autores utilizam um método de reparo de Chang *et al.* (2000) garantindo que a cadeia de indivíduos sempre seja viável, respeitando as restrições de orçamento, cardinalidade e restrições de limite inferior/superior.

Dada a natureza do algoritmo genético, as operações de crossover e mutação podem desrespeitar algumas restrições. Para corrigir esses elementos, é utilizado um mecanismo de reparação que consiste na geração de um novo vetor que respeita as restrições do modelo. Além dos operadores, o algoritmo é modificado de forma que consegue resolver o problema sem desrespeitar as restrições.

Em seus experimentos, os autores utilizaram uma base de dados clássica, apresentada inicialmente por Chang *et al.* (2000). Esses dados são os preços semanais dos índices de alguns países, são eles: HangSeng (Hong Kong), Dax 100 (Alemanha), FTSE 100 (Reino Unido), S&P

(Estados Unidos) e Nikkei 225 (Japão) no período de março de 1992 até setembro de 1997.

Para analisar a perfomance da solução implementada, os autores traçaram a *fronteira eficiente*, que corresponde a uma curva traçada nos eixos X e Y, em que X é o retorno e o Y o risco. Além disso, também foi utilizado como parâmetro de comparação o menor erro percentual médio, onde a solução de Jalota (2016) também teve seus resultados comparados com outros algoritmos de diferentes abordagens, como PSO, Simulating Annealing(SA), Busca Tabu(TS) e outro GA. Conforme o índice considerado, a solução proposta pelos autores destacava-se como superior em relação aos outros algoritmos comparados ou figurava entre as melhores. Ou seja, no geral, apresentou um bom comportamento ao considerar o *erro percentual médio*.

4.2 A Novel Particle Swarm Optimization for Portfolio Optimization Based on Random Population Topology Strategies

Yin *et al.* (2015) adota um modelo generalizado do modelo de Média-Variância (MV) padrão, incorporando restrições de cardinalidade e limites, semelhante ao proposto por Chang *et al.* (2000). Além disso, utiliza as mesmas instâncias clássicas que Jalota (2016) para a análise dos resultados.

Neste estudo, os autores propõem diferentes abordagens para o PSO, utilizando quatro algoritmos distintos, mas inter-relacionados, para topologia. Esses algoritmos combinam abordagens aleatórias com dinâmicas. Embora o PSO consiga lidar com as restrições do problema, os autores não detalham as abordagens ou técnicas específicas utilizadas para isso.

Yin *et al.* (2015) oferece uma análise abrangente dos resultados, empregando diversos parâmetros para comparação com outros algoritmos, além de aplicar diferentes algoritmos de topologia. Os parâmetros considerados incluem a *distância euclidiana média*, a *variância do erro de retorno*, o *erro médio de retorno* (conforme definido em Cura (2009)) e o *tempo de execução*.

Adicionalmente, foi realizada uma análise gráfica detalhada utilizando a *fronteira eficiente* como referência. Além da análise individual de cada índice, o autor faz comparações com a fronteira padrão e com as de outros algoritmos. Com base nos resultados apresentados, os autores concluem que as estratégias adotadas podem ser métodos promissores e significativos para a melhoria dos algoritmos PSO, e que os algoritmos apresentados são soluções eficazes para o Problema de Otimização de Portfólio Financeiro com restrição de cardinalidade.

4.3 Síntese dos Trabalhos Correlatos

Os trabalhos apresentados nas seções anteriores desempenharam papel essencial para o desenvolvimento desta pesquisa, fornecendo uma base fundamental para compreensão do problema de Otimização de Portfólio. Tendo isso em vista, o intuito não é trazer uma base comparativa com os resultados desses trabalhos, mas auxiliar na identificação de variáveis relevantes, métricas e procedimentos adotados. Em particular, foram uma inspiração, principalmente para o desenvolvimento de métodos como o de reparação e as abordagens adotadas na análise dos resultados.

5 ABORDAGENS UTILIZADAS: GENETIC ALGORITHM(GA) E PARTICLE SWARM OPTIMIZATION(PSO)

Como visto no capítulo 2, o GA e o PSO, são metaheusrísticas inspiradas nos princípios da seleção natural, dado suas características e versatilidade, na literatura é evidenciada performances exitosas quando aplicadas ao Problema de Otimização de Portfólio. No entanto, para serem empregadas é necessária atenção às restrições do problema e aos ajustes de suas operações. O presente trabalho implementou dois algoritmos modificados, como mostrado a seguir, para o modelo de restrições (3.5) - (3.9).

5.1 Genetic Algorithm(GA)

Os procedimentos adotados no GA são realizados de forma iterativa e sequencial, com um número máximo de execuções definido (*maxIteracoes*). A sequência segue-se em ordem de:

1. Selecionar indivíduos para se reproduzirem,
2. Realizar operações de crossover e mutação,
3. E substituir a população por a nova gerada, podendo usar algumas abordagens como elitismo.

No presente trabalho, o GA (Algoritmo 7) foi implementado seguindo esse processo, com modificações para que os operadores de *crossover* e *mutação* respeitassem as restrições de cardinalidade e aos limites inferiores e superiores. Além disso, para garantir a viabilidade completa das soluções, foi utilizado, inspirando-se no trabalho de Jalota e Thakur (2018), uma abordagem de reparação. As funções utilizadas serão explicadas e seus algoritmos apresentados a seguir.

Na etapa de seleção o objetivo é selecionar os indivíduos com melhores aptidões, nesse trabalho, foi implementado a seleção por roleta. Conforme Holland (1992), a seleção por roleta corresponde a selecionar os indivíduos com base em suas probabilidades que são proporcionais ao seu valor de aptidão. Como o próprio nome revela, o algoritmo pode-se relacionar com uma roleta dividida em partes, onde as maiores partes são destinadas aqueles indivíduos com maior probabilidade de serem escolhidos e as menores representam aqueles com menor probabilidade de escolha (Shukla *et al.* (2015)).

Para realizar a seleção, o algoritmo recebe dois parâmetros, *fitness* e *tamPop*, que representam a matriz com as aptidões de todos os indivíduos e o tamanho da população, respec-

tivamente. O cálculo da seleção está representado nas linhas 1 e 2, com $fitness = (fitness - fitness.min())/(fitness.max() - fitness.min() + 1^{-10}) + 0.2$ e $prob = fitness/fitness.soma()$, sendo a única diferença da seleção padrão, a soma de 1^{-10} e 0.2. Esses valores estão presentes no cálculo para garantir que mesmo os piores indivíduos, com baixas aptidões, possam ser escolhidos pois eles podem ter valores muito próximos de zero. A escolha de um indivíduo "ruim"pode ser justificada para contribuir com a diversidade genética e exploração do espaço de busca. O pseudocódigo correspondente está representado em Algorítimo 3.

Algoritmo 3 : Função de Seleção Por Roleta

Entrada: $fitness, tamPop$

1: $fitness = (fitness - fitness.min())/(fitness.max() - fitness.min() + 1^{-10}) + 0.2$
2: $prob = fitness/fitness.soma()$
3: $selecionadosIndex = selecionar(tamPop, prob)$
4: **retorna** $X(selecionadosIndex)$

Fonte: Próprio autor.

A etapa de crossover é realizada após a seleção, como mostrado anteriormente na seção 2.3.2 esta possui como objetivo combinar dois indivíduos, denominados de *pais* afim de produzir novos indivíduos denominados *filhos* com maior variedade genética.

Existem várias formas de implementar esta etapa, nesse trabalho, foi utilizado uma variação de um crossover uniforme com *crossover de um ponto*. O primeiro se caracteriza por selecionar os genes dos pais, de forma independente, com base em uma probabilidade, no caso, de 50% e o segundo é a forma tradicional do *crossover* mostrada na seção 2.3.2.

O algoritmo 5, apresentado a seguir, recebe X e $tamPop$ que representam a população e tamanho da população e gera dois arrays filhos a partir de dois arrays de pais, sendo o primeiro filho obtido da escolha aleatória de índices dos pais e o segundo dos índices restantes.

Algoritmo 4 : Função de Crossover

Entrada: $X, tamPop$

1: **para** $i = 0$ *em* $tamPop$ **faça**
2: $pai1 = X(i)$
3: $pai2 = X(i+1)$
4: Escolhe aleatoriamente 5 índices de $pai1$
5: Escolhe aleatoriamente 5 índices de $pai2$
6: Atribui os elementos escolhidos anterioremente para $filho1$
7: Atribui os elementos não escolhidos dos pais para $filho2$
8: $i = i+2$
9: **fim para**
10: **retorna** Array de filhos

Fonte: Próprio autor.

Responsável por introduzir variedade genética, a fase de mutação permite maior exploração do algoritmo, modificando algumas partes de um indivíduo. O algoritmo de mutação implementado, utiliza uma abordagem conhecida ao utilizar valores reais, a de incremento e decremento de genes, porém modificado para não alterar o limite superior estabelecido. No pseudocódigo abaixo, é mostrado os passos dessa mutação, onde recebe como entrada os já conhecidos X e $tamPop$ e os novos elementos, $tamDim$ que representa a dimensão do cromossomo e $probMutacao$ que como o nome revela, é a probabilidade dessa mutação ocorrer. Esse valor de $probMutacao$, assim como o *rand* está no intervalo de 0 e 1, portanto, a comparação entre eles na linha 3 é válida.

Um conceito fundamental relacionado ao uso das probabilidades no algoritmo é a distribuição uniforme, que se aplica tanto à função *rand* na linha 3 (com intervalo entre 0 e 1) quanto à linha 5 (onde o intervalo é definido por $limiteInferior$ e $elementoRandomico$). Se caracteriza por ser uma distribuição de probabilidade onde todos os resultados dentro do intervalo considerado, possuem a mesma chance de ocorrer, isto é, não possui um viés ou tendência à qualquer valor em específico. Por esses fatores, tende a ser muito utilizado em algoritmos genéticos.

Como temos a restrição de cardinalidade a ser considerada, a mutação não pode ocorrer em qualquer elemento, por isso, j precisa ser diferente de zero e o mesmo ocorre para $elementRandomico$. Já para a escolha do valor a ser incrementado e decrementado, representado por $delta$, é feita de forma randômica e uniforme, considerando a restrição de limite inferior e o valor do outro gene randômico escolhido. Caso a soma respeite o limite superior, o valor de $delta$ permanece o inicialmente escolhido, caso contrário, ele é modificado, passando a ser

resultado de $limiteSuperior - delta$. Pontua-se que alterar o valor de $delta$ é necessário para garantir que a mudança ocorra, porém, assegurando o limite superior e a relação com o valor randômico de $delta$ escolhido inicialmente. Esse processo está descrito entre as linhas 4 e 13.

Algoritmo 5 : Função de Mutação

Entrada: X, $probMutacao$, $tamPop$, $tamDim$

1: **para** $i = 0$ *em* $tamPop$ **faça**
2: **para** $j = 0$ *em* $tamDim$ **faça**
3: **se** $rand(0,1) < probMutacao$ *e* $j \neq 0$ **então**
4: Escolhe outro elemento, $elementoRandomico$, de forma randômica
5: $delta = rand(limiteInferior,\ elementoRandomico)$
6: **se** $X[i][j] + delta < limiteSuperior$ **então**
7: Adiciona delta em $X[i][j]$
8: Remove delta de $X[i][elementoRandomico]$
9: **fim se**
10: **se** $X[i][j] + delta > limiteSuperior$ **então**
11: Modifica delta para ser $limiteSuperior - delta$
12: Adiciona $delta$ em $X[i][j]$
13: Remove $delta$ de $X[i][elementoRandomico]$
14: **fim se**
15: **fim se**
16: **fim para**
17: **fim para**
18: **retorna** X com valores modificados

Fonte: Próprio autor.

Percebe-se que após a realização dos operadores, as restrições são desrespeitadas. O *crossover*, com seu processo natural dos pais, resulta em filhos que infringem a restrição de orçamento. A *mutação*, por outro lado, garante somente que os limites superiores sejam respeitados. Visando reparar esses indivíduos, é usada uma função de reparação (Algoritmo 6) que verifica essas restrições e realiza ajustes quando necessário. Esses ajustes consistem em acrescentar um valor aos ativos que não respeitam a restrição de limite inferior. Esse o valor acrescido é tirado de um outro ativo, desde que a restrição continue sendo respeitada (linha 3). Para garantir a restrição de orçamento, a normalização é utilizada (linha 7).

Algoritmo 6 : Função de reparação

Entrada: X

1: **para** $i = 0$ *em* X **faça**
2: **se** $i < limiteInferior$ **então**
3: $X = repararLimitesInferiores(X)$
4: **fim se**
5: **fim para**
6: **se** $\sum_{i=1}^{n} X \neq 1$ **então**
7: $X = normalizar(X)$
8: **fim se**

Pontua-se que o procedimento de reparação, possui a funcionalidade de normalização, como disposto no pseudo-código. Isso ocorre porque, embora os procedimentos de *crossover* e *mutação* tenham apresentado, de forma geral, bons comportamentos na garantia da manutenção dos limites estabelecidos, por vezes suas operações aleatórias podem desrespeitar as restrições impostas. Dessa forma, o procedimento de Jalota e Thakur (2018) contribuiu para inspirar a elaboração do procedimento apresentado, que embora simples, possui funcionalidade importante.

Dados a apresentação de todos os elementos principais, o pseudocódigo 7 mostra o funcionamento geral do algoritmo genético, seguindo os passos tradicionais, com adição da função de reparação e elitismo. Sendo esse último, uma técnica com intuito de preservar os melhores da geração. Há diferentes formas de aplicar o elitismo, nesse trabalho, foi adotado um elitismo simples com elementos de uma abordagem competitiva, onde "pais"e a nova população (formada após aplicar seleção, crossover e mutação) se juntam e aqueles com melhores aptidões são selecionados. Podemos ver o elitismo ocorrendo nas linhas 3, 9 e 10.

Algoritmo 7 : Algoritmo Genético

Entrada: $maxIteracoes$

1: $X = inicializar(X)$
2: **para** $i = 0$ *em* $maxIteracoes$ **faça**
3: Selecionar indivíduos
4: $pais = X$
5: Fazer operação de Crossover
6: Usar Função de Reparação
7: Fazer operação de Mutação
8: Usar Função de Reparação
9: Preservar o melhor usando elitismo, comparando o novo X com *pais*
10: $X = melhores(X)$
11: Guarda o melhor da geração
12: **fim para**
13: **retorna** O melhor de todas as gerações

5.2 Particle Swarm Optimization(PSO)

O algoritmo PSO, como mencionado na Subseção 2.3.3.2, é inspirado no comportamento de enxames de partículas. Neste processo, as partículas se movem pelo espaço de busca de maneira colaborativa. No algoritmo, esse processo ocorre de forma iterativa, onde em cada iteração, cada partícula atualiza sua posição (logo, muda de local no espaço) com base na sua melhor solução encontrada até o momento, chamada de *pbest* e na melhor solução entre todas as partículas (solução global), chamada de *gbest*. Dessa forma, todas as partículas se atualizam a cada iteração, buscando se aproximar das soluções ótimas. No presente trabalho, o PSO implementado (Algoritmo 8) também pode ser chamado por *WPSO*, pois é adicionado ao PSO um parâmetro de peso de inércia w no cálculo da velocidade de uma partícula, na perspectiva de balancear a busca global e local (Shi e Eberhart (1999)). Dessa forma, a Equação do PSO, se torna:

$$v_{id} = w_i * v_{id} + c_1 * rand() * (p_{id} - x_{id}) + c_2 * rand() * (p_{gd} - x_{id}) \tag{5.1}$$

O Algoritmo 9 apresenta a função de atualização da posição utilizada no *PSO com peso de inércia w*. As variáveis mostradas fazem parte da fórmula padrão do algoritmo, onde, para cada elemento i, v_i é a velocidade da partícula, x_i é sua posição, *pBest* é a melhor posição anterior da partícula e *gBest* é a melhor posição conhecida de todas as partículas. O valor de $rand()$ é uma função aleatória entre $[0,1]$, e c_1 e c_2 são duas constantes de aceleração referentes ao melhor valor individual e global, respectivamente. Essa função de atualização possui a adaptação necessária para lidar com a restrição de cardinalidade, pois o cálculo é aplicado apenas aos elementos já escolhidos. Para às outras restrições, o processo de reparação (Algoritmo 6), apresentado no algoritmo genético, também é utilizado. Esse uso torna-se possível pois apesar das diferenças entre as abordagens do GA e PSO, o indivíduo possui a mesma estrutura e precisam respeitar, obviamente, as mesmas restrições.

O Algoritmo 8 mostra o funcionamento geral do *WPSO* implementado. Na linha 1, temos a atualização da posição das partículas, que foi detalhada no parágrafo anterior. Nas linhas 5, 6 e 7, temos os passos de atualização de pbest e gbest que serão guardados para serem utilizados na atualização da posição. Voltando para a linha 4, temos a função de reparação que ajusta todos os indivíduos a fim de fazê-los respeitar todas as restrições. Por fim, o retorno do algoritmo será o último *gbest* encontrado no espaço de busca que representa a melhor solução.

Algoritmo 8 : PSO com peso de inércia w

1: **Inicialize** a população de partículas
2: **para** $i = 0$ *em maxIteracoes* **faça**
3: Avaliar o valor de fitness em X
4: Atualizar pbest para cada partícula
5: Atualizar gbest
6: *atualizarPosicao(X)*
7: Usar Função de Reparação
8: **fim para**
9: **retorna** *gBest* como a melhor solução encontrada

Algoritmo 9 : Atualização da posição

Entrada: X
1: **para** $i = 0$ *em* X **faça**
2: **se** $i \neq 0$ **então**
3: Atualize a velocidade $v[i]$:
4: $v[i] = w \cdot v[i] + c1 \cdot rand() \cdot (pBest[i] - x[i]) + c2 \cdot rand() \cdot (gBest - x[i])$
5: Atualize a posição $x[i]$:
6: $x[i] = x[i] + v[i]$
7: **fim se**
8: **fim para**

Entendido o funcionamento dos algoritmos implementados nesse trabalho, a seguir, no Capítulo 6, serão apresentados os parâmetros adotados para a realização dos experimentos computacionais, as discussões sobre os resultados com uma análise detalhada de cada um deles e as evidências colhidas.

6 EXPERIMENTOS E RESULTADOS

Este capítulo apresenta os resultados computacionais obtidos dos experimentos realizados na presente pesquisa. Nele é detalhado o processo de análise e comparação dos comportamentos computacionais pela combinação de três fatores: instâncias, tamanho da população e quantidade de iterações dos algoritmos.

6.1 Dados Experimentais - Instâncias

As entradas para as heurísticas na área da Otimização Computacional são denominadas por *instâncias*, e as utilizadas nesta pesquisa correspondem aos índices das bolsas de diferentes países: Hong Kong, Alemanha, Reino Unido, Estados Unidos e Japão. As informações consideram os preços semanais do período de março de 1992 a setembro de 1997.

A justificativa para o uso dessas instâncias se deve à representatividade para a área e à solidez para a verificação de desempenho dos algoritmos. Os trabalhos correlatos de Jalota e Thakur (2018), Yin *et al.* (2015) e Chang *et al.* (2000), citados no Capítulo 4, utilizam em seus experimentos estes mesmos dados.

6.2 Métricas

As métricas utilizadas nesse processo de análise foram a *fronteira de eficiência*, *média da distância euclidiana*, *erro médio de retorno*, *erro médio da variância* e *tempo de execução* dos experimentos. Todas as métricas, com exceção da *fronteira de eficiência*, foram incluídas no processo de execução do algoritmo, ou seja, são calculadas durante os experimentos.

Justifica-se o uso da *fronteira eficiente* devido a caracterização dada por Markowitz como o conjunto de carteiras que possuem o maior retorno dado um nível de risco. Essa fronteira é representada através de uma curva, onde o eixo X representa o risco e o Y o retorno. Sendo comumente utilizada em pesquisas na área em que abordam o Problema de Portfólio por permitir o comportamento do valor obtido na função objetivo, como visto no Capítulo 4.

As demais métricas possuem seus usos justificados devido à diferentes aspectos. O *erro médio do retorno* é a média das diferenças entre os retornos reais e os retornos esperados. O *erro médio da variância* possibilita avaliar quanto a variância prevista difere da variância real dos retornos, métrica essa que está relacionada ao risco. A *distância euclidiana* permite calcular a distância entre os pontos da fronteira eficiente padrão com os da fronteira eficiente obtidas. O

tempo, medido em segundos, representa o tempo de execução de cada metaheurística para cada instância.

6.3 Cenário

Os experimentos computacionais foram executados em uma máquina pessoal com processador Intel Core i5-8265U de 0.5 GHz e 4 núcleos de processamento, 8 GB de memória RAM com frequência de 2400 MT/s e sistema operacional Linux Mint. Os algoritmos foram desenvolvidos em linguagem Python® versão 3.7.6.

A carga de execução dos experimentos correspondeu a uma sequência única, e o tamanho da população e a quantidade de iterações foram estabelecidos empiricamente, escolhidos após alguns pré-testes. Foram definidos os valores de 50, 80 e 100 para a população, e a quantidade de iterações, em 100, 500 e 1000. As instâncias mencionadas na seção 6.1 foram as Hang Seng de Hong Kong, DAX 100 da Alemanha, FTSE 100 do Reino Unido, S&P 100 dos EUA e Nikkei 225 do Japão.

Alguns parâmetros são importantes de serem destacados. Para o algoritmo PSO a topologia adotada foi a global e a a aceleração com que os indivíduos realizavam a trajetória, foi definida como $0,5$; e os pesos, estabelecidos como $0,8$. Para o GA o processo de seleção foi determinado por meio da seleção por roleta e a probabilidade do processo de mutação foi definida como $0,2$. Ressalta-se que em vista dos trabalhos de Yin *et al.* (2015) e Cura (2009) não mencionarem estes valores, a presente pesquisa realizou estas escolhas por meio de análise empírica.

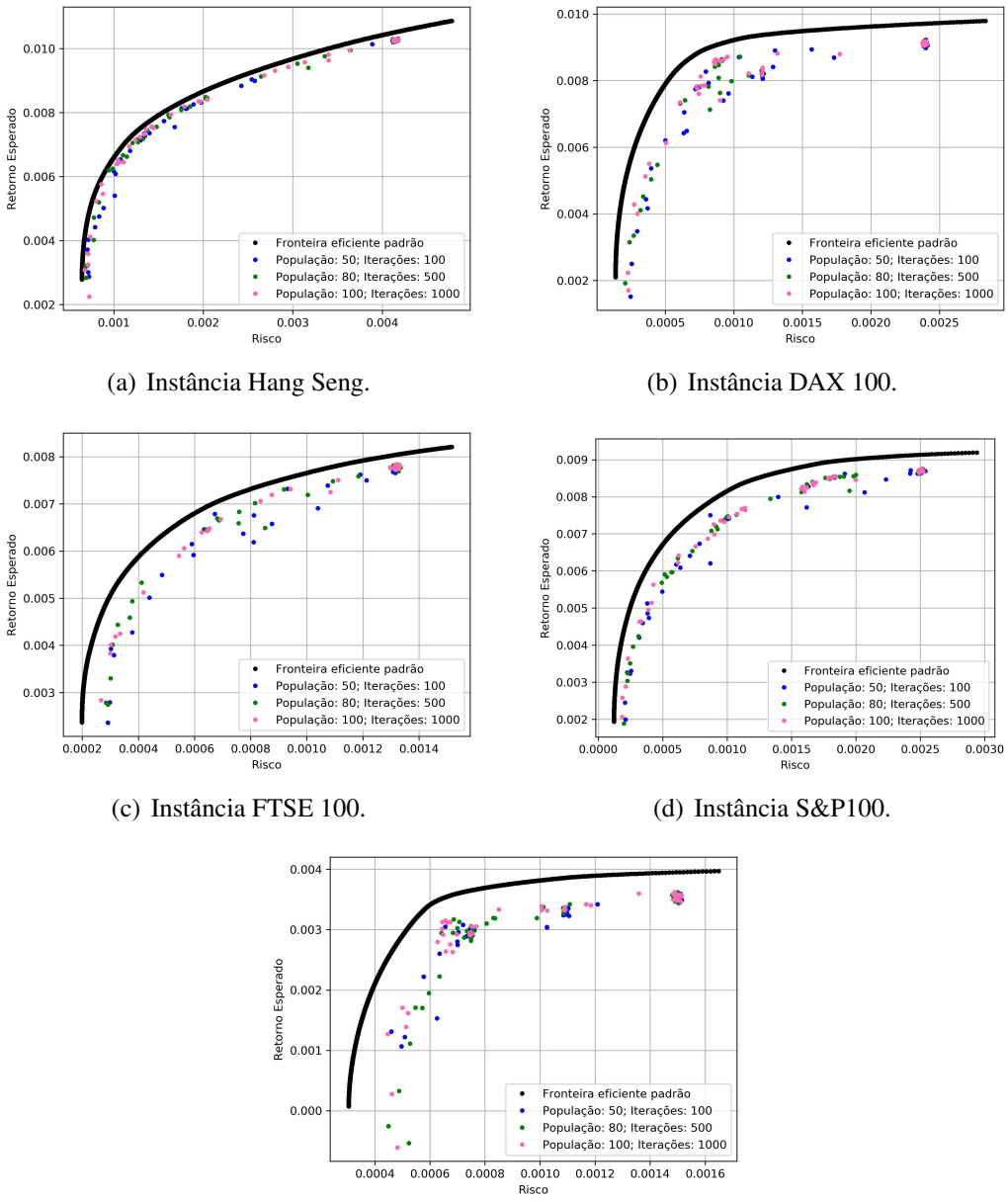
6.4 Resultados e Discussões

Os desempenhos foram analisados considerando a variação dos parâmetros e utilizando as métricas apresentadas na Seção 6.2, em três etapas: a primeira análise apresenta o comportamento individual de cada metaheurística por meio da *fronteira eficiente*; a segunda análise parte da verificação dos desempenhos dos algoritmos frente às métricas *média da distância euclidiana*, *erro médio de retorno*, *erro médio da variância* e *tempo de execução* dos experimentos; a terceira e última etapa de análise corresponde à comparação dos melhores desempenhos obtidos das duas metaheurísticas por meio da *fronteira eficiente*.

Ao analisar o desempenho individual da estratégia PSO (Figura 6) percebe-se que,

no geral, os testes em que os parâmetros da população foi estabelecido como 100 e o número de iterações como 1000, o algoritmo apresentou as melhores performances com maior proximidade a fonteira eficiente padrão, revelando assim, maior capacidade de sair de ótimos locais ao comparar com os demais comportamentos. Apesar dos testes realizados com as instâncias DAX 100 (Figura 7(c)) e FTSE 100 (Figura 7(d)), em que o algoritmo obteve uma solução não tão boa. Cabe destacar que para a instância Hang Seng (Figura 7(a)), o desempenho do PSO mostra equivalência em todos os parâmetros utilizados.

Figura 6 – Desempenho do PSO com variação dos parâmetros do tamanho da população e da quantidade de iterações.



(a) Instância Hang Seng.

(b) Instância DAX 100.

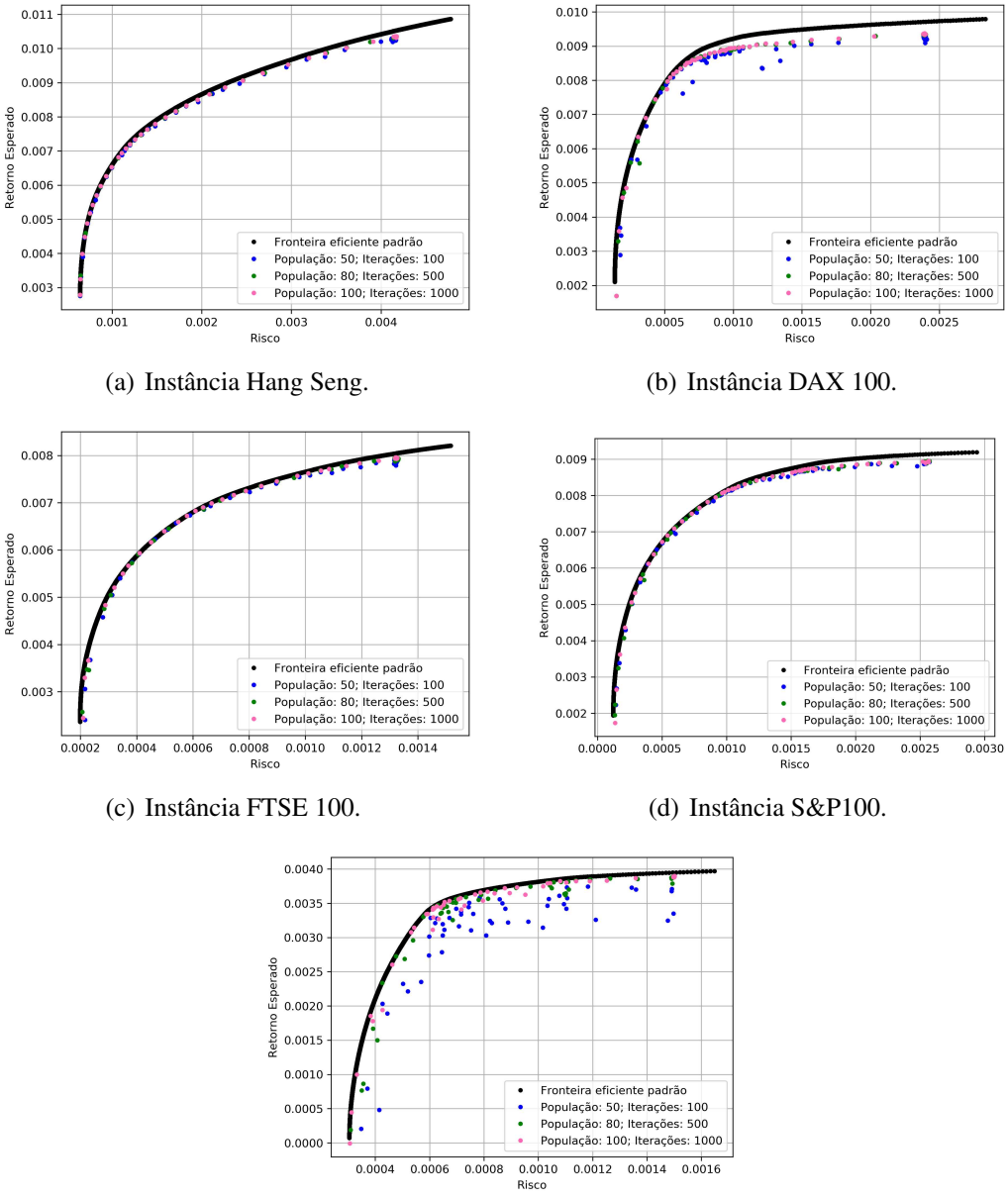
(c) Instância FTSE 100.

(d) Instância S&P100.

(e) Instância Nikkei.

Ainda na perspectiva da análise individual, utilizando o mesmo conjunto de parâmetros, foi analisado o desempenho do algoritmo GA (Figura 7). Observa-se que, semelhante aos testes do PSO, o comportamento do GA obteve melhor desempenho quando utilizado com os parâmetros de 100 indivíduos na população e 1000 execuções. Contudo, diferentemente do PSO, para as demais variações de parâmetros, o GA apresentou boa performance.

Figura 7 – Desempenho do GA com variação dos parâmetros do tamanho da população e da quantidade de iterações.



(a) Instância Hang Seng.

(b) Instância DAX 100.

(c) Instância FTSE 100.

(d) Instância S&P100.

(e) Instância Nikkei.

A segunda fase da análise compreendeu na observação do desempenho do GA e do PSO com a variação dos parâmetros do tamanho da população, em 50, 80 e 100, e da quantidade

de iterações, em 100, 500 e 1000, resultando em 60 casos comparativos. Considerando os valores de *tempo de execução*, percebe-se que o GA possui índices mais elevados no consumo de tempo; dos 15 casos comparativos, em 11 o PSO obteve melhor desempenho. Em relação à métrica de *distância euclidiana*, o GA obteve melhor desenvoltura em todos os casos de comparação para todas as instâncias. Quanto às demais métricas, *erro médio do retorno* e *erro médio da variância*, os desempenhos foram equivalentes: o GA obteve 7 e 8 ganhos, respectivamente, enquanto o PSO obteve 8 e 7.

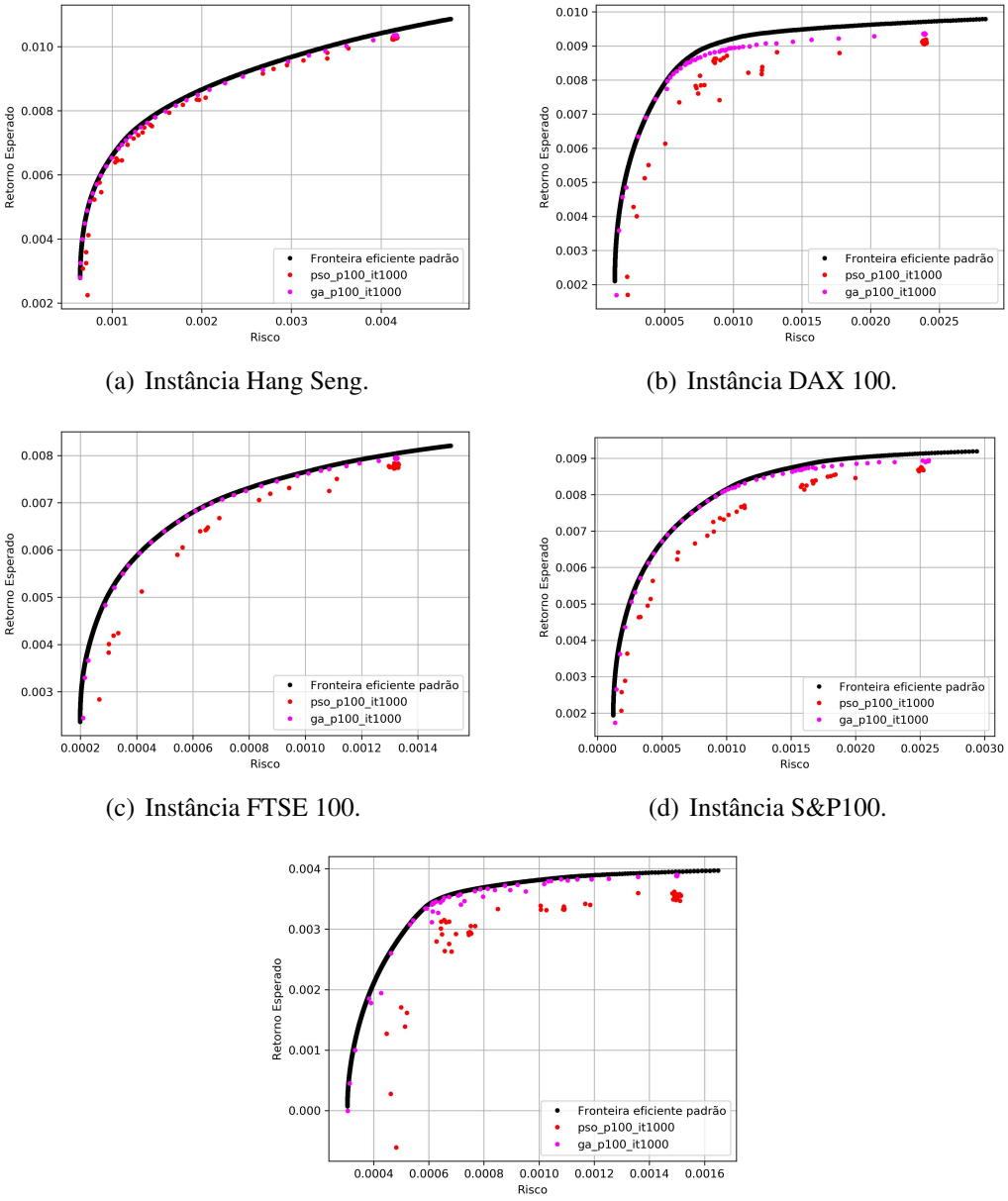
Tabela 1 – Desempenho do GA e do PSO por meio das métricas de erro médio do retorno, erro médio da variância, distância euclidiana e tempo.

Instâncias	Métricas	Hang Seng	DAX 100	FTSE 100	SP 100	Nikkei
GA_p50_it100	Distância Euclidiana	0.000110	0.000248	0.0000691	0.000103	0.000199
	Erro médio de retorno	45.083	95.702	29.050	50.109	841.776
	Erro médio da variância	122.938	709.601	247.421	221.539	841.776
	Tempo(s)	286.599	1374.9	30716.1	32050.97	8449.06
GA_p80_it500	Distância Euclidiana	0.000080	0.000172	0.000048	0.000083	0.000056
	Erro médio de retorno	32.6245	94.9938	20.8628	41.3317	56.2702
	Erro médio da variância	87.7273	419.699	171.302	176.9161	221.109
	Tempo(s)	2389.9	10812.42	11175.40	14840.9	64590.7
GA_p100_it1000	Distância Euclidiana	0.000080	0.000166	0.0000421	0.000083	0.0000426
	Erro médio de retorno	0.630366	1.818650	0.356857	1.007747	-20.024100
	Erro médio da variância	1.686624	7.795563	2.829000	2.97404	3.12332
	Tempo(s)	6716.76	27114.60	31973.79	33190.32	162577.6
PSO_p50_it100	Distância Euclidiana	0.00017	0.00048	0.00019	0.00035	0.00034
	Erro médio de retorno	62.6494	195.606	66.5584	163.118	299.797
	Erro médio da variância	332.764	1522.37	891.541	970.924	1319.39
	Tempo(s)	167.882	637.026	641.284	755.875	3833.53
PSO_p80_it500	Distância Euclidiana	0.000154	0.000449	0.000172	0.0003390	0.000327
	Erro médio de retorno	55.8564	165.925	62.2187	152.449	51.2473
	Erro médio da variância	263.812	1336.24	774.095	946.766	1107.89
	Tempo(s)	1490.51	5380.41	5850.32	6270.16	25945.9
PSO_p100_it1000	Distância Euclidiana	0.00016	0.00040	0.00016	0.00033	0.00030
	Erro médio de retorno	1.56595	3.51747	1.20040	2.83368	3.69956
	Erro médio da variância	4.99898	22.2750	14.2308	18.1501	19.2928
	Tempo(s)	3742.33	13301.9	14278.6	17210.7	67639.4

Observando que os algoritmos obtiveram os melhores desempenhos quando utilizados os parâmetros de população igual a 100 e o número de iterações igual a 1000, e que o GA obteve melhor aproximação da fronteira eficiente padrão. Buscou-se na terceira fase de análise verificar suas comparações com os mesmos parâmetros através da fronteira de eficiência. Ao analisar a Figura 8, percebe-se um melhor desempenho do GA que se aproxima da fronteira eficiente padrão. Destaca-se, porém, a instância Nikkei (Figura 9(e)), que possui o maior número de dados, percebe-se uma dificuldade do GA de sair de ótimos locais, apresentando assim uma performance inferior quando comparado aos seus resultados nas outras instâncias. Contudo,

apesar de ser uma solução não muito distante da fronteira eficiente padrão, ainda se mantém melhor que o PSO.

Figura 8 – Comparação da curva de eficiência do PSO e do GA com 100 indivíduos em 1000 iterações.



(a) Instância Hang Seng.

(b) Instância DAX 100.

(c) Instância FTSE 100.

(d) Instância S&P100.

(e) Instância Nikkei.

A equivalência, por vezes, é uma característica evidenciada entre os desempenhos das meta-heurísticas na área da Otimização Computacional, dependendo da natureza dos procedimentos e do problema tratado, conforme foi comprovado nas análises realizadas na primeira e segunda etapa desta seção. Contudo, foi importante observar a importância das etapas dois e três em que a análise do desempenho das estratégias metaheurísticas por meio da *fronteira eficiente*

demonstrou algo já apontado pela *distância euclidiana*: a verificação de um comportamento superior do GA em relação ao PSO.

7 CONCLUSÃO

O presente trabalho teve como objetivo verificar o desempenho individual e comparativo das metaheurísticas clássicas *Particle Swarm Optimization*(PSO) e *Genetic Algorithm*(GA) na resolução do Problema de Otimização de Portfólio, possuindo como variação para a carga de avaliação o tamanho da população, o número de iterações dos algoritmos e as instâncias utilizadas.

A justificativa para a escolha do uso de metaheurísticas em detrimento as demais técnicas existentes na área da Otimização Computacional, ocorreu devido a natureza do problema trabalhado. Sobre as metaheurísticas abordadas o PSO pertence a uma classe de metaheurística baseada no comportamento de grupo, inspirada na evolução biológica da natureza. Já a metaheurística GA pertence a uma classe inspirada no processo da evolução natural, baseada nos princípios de seleção natural e genética.

Para a condução dos experimentos foram utilizadas instâncias e métricas consolidadas na literatura do Problema, que geraram 75 casos de testes. Em que 60 casos foram advindos da comparação das metaheurísticas quanto as métricas de *média da distância euclidiana*, *erro médio de retorno*, *erro médio da variância* e *tempo de execução* dos experimentos, quando analisados os tamanho da população (em 50, 80 e 100 indivíduos) e o número de iterações (100, 500 e 1000), e 15 casos advindos da comparação das metaheurósticas quanto a métrica da *fronteira de eficiência*, quando analisados os comportamentos de cada algoritmo à variação do tamanho da população (em 50, 80 e 100 indivíduos) e o número de iterações (100, 500 e 1000).

Os resultados computacionais revelaram que o melhor desempenho individual dos algoritmos foi obtido ao utilizar um número de população igual a 100 indivíduos, executando 1000 iterações. Ao comparar o desempenho das metaheurísticas frente a todas as métricas foi constatado um desempenho promissor do algoritmo GA, que possuiu melhor desempenho comparado ao PSO.

Desta forma, conclui-se que as contribuições advindas desta pesquisa favorecem a literatura através da análise da performance de um algoritmo inspirado nos princípios da seleção natural e de adaptação genética utilizando parâmetros e procedimentos já existentes na literatura.

Buscando tornar os algoritmos mais complexos para analisar a perfomance do PSO, em trabalhos futuros, pode-se implementar novas formas de lidar com as restrições em substituição as que foram implementadas nesse trabalho. Para o PSO, podem ser utilizadas também topologias dinâmicas para analisar o impacto no resultado. Além disso, cabe-se analisar

a perfomance de outros algortimos, tais como *Tabu Search*, *Binary Genetic Algorithm* e *Artificial Bee Colony* que podem proporcionar estudos interessantes para a área.

REFERÊNCIAS

BENTO, E. P.; KAGAN, N. Algoritmos genéticos e variantes na solução de problemas de configuração de redes de distribuição. **Sba: Controle & Automação Sociedade Brasileira de Automatica**, SciELO Brasil, v. 19, n. 3, p. 302–315, 2008.

BLUM, C.; ROLI, A. Hybrid metaheuristics: an introduction. In: **Hybrid Metaheuristics**. [S.l.]: Springer, 2008. p. 1–30.

CHANG, T.-J.; MEADE, N.; BEASLEY, J. E.; SHARAIHA, Y. M. Heuristics for cardinality constrained portfolio optimisation. **Computers & Operations Research**, Elsevier, v. 27, n. 13, p. 1271–1302, 2000.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to Algorithms**. 3rd. ed. Cambridge, MA: MIT Press, 2009.

CRAMA, Y.; SCHYNS, M. Simulated annealing for complex portfolio selection problems. **European Journal of operational research**, Elsevier, v. 150, n. 3, p. 546–571, 2003.

CURA, T. Particle swarm optimization approach to portfolio optimization. **Nonlinear analysis: Real world applications**, Elsevier, v. 10, n. 4, p. 2396–2406, 2009.

DARWIN, C. *et al.* **On the origin of species by means of natural selection or the preservation of favoured races in the struggle for life.** [S.l.]: Books, Incorporated, Pub., 1859. v. 2.

ESTRADA, J. Mean-semivariance behavior: Downside risk and capital asset pricing. **International Review of Economics & Finance**, Elsevier, v. 16, n. 2, p. 169–185, 2007.

GAMARRA, C.; GUERRERO, J. M. Computational optimization techniques applied to microgrids planning: A review. **Renewable and Sustainable Energy Reviews**, Elsevier, v. 48, p. 413–424, 2015.

HERTZ, A.; WIDMER, M. Guidelines for the use of meta-heuristics in combinatorial optimization. **European Journal of Operational Research**, Elsevier, v. 151, n. 2, p. 247–252, 2003.

HOLLAND, J. H. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence**. [S.l.]: MIT press, 1992.

JALOTA, H. **Soft computing approaches for portfolio optimization: an empirical study (PhD)**. Tese (Doutorado) — IITMandi, 2016.

JALOTA, H.; THAKUR, M. Genetic algorithm designed for solving portfolio optimization problems subjected to cardinality constraint. **International Journal of System Assurance Engineering and Management**, Springer, v. 9, n. 1, p. 294–305, 2018.

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

KENNEDY, J. The particle swarm: social adaptation of knowledge. In: IEEE. **Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)**. [S.l.], 1997. p. 303–308.

KIM, H.-j.; SHIN, K.-s. A hybrid approach based on neural networks and genetic algorithms for detecting temporal patterns in stock markets. **Applied Soft Computing**, Elsevier, v. 7, n. 2, p. 569–576, 2007.

KONNO, H.; YAMAZAKI, H. Mean-absolute deviation portfolio optimization model and its applications to tokyo stock market. **Management science**, INFORMS, v. 37, n. 5, p. 519–531, 1991.

LINDEN, R. **Algoritmos genéticos (2a ediçao)**. [S.l.]: Brasport, 2008.

LUENBERGER, D. G.; YE, Y. *et al.* **Linear and nonlinear programming**. [S.l.]: Springer, 1984. v. 2.

MANSINI R., O. W.; SPERANZA, M. **Linear and Mixed Integer Programming for Portfolio Optimization**. S.l.: Springer International Publishing, 2015.

MARKOWITZ, H. Portfolio selection. **The Journal of Finance**, v. 7, n. 1, p. 77–91, mar. 1952. Disponível em: <https://www.jstor.org/stable/2975974>.

MISHRA, S. K. Robust and constrained portfolio optimization using multiobjective evolutionary algorithms. In: . [S.l.: s.n.], 2012.

MORAL-ESCUDERO, R.; RUIZ-TORRUBIANO, R.; SUÁREZ, A. Selection of optimal investment portfolios with cardinality constraints. In: IEEE. **2006 IEEE International Conference on Evolutionary Computation**. [S.l.], 2006. p. 2382–2388.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. **Combinatorial optimization: algorithms and complexity**. [S.l.]: Courier Corporation, 1998.

REEVES, C.; ROWE, J. E. **Genetic algorithms: principles and perspectives: a guide to GA theory**. [S.l.]: Springer Science & Business Media, 2002. v. 20.

ROCKAFELLAR, R. T.; URYASEV, S. *et al.* Optimization of conditional value-at-risk. **Journal of risk**, Citeseer, v. 2, p. 21–42, 2000.

ROMAN, D.; DARBY-DOWMAN, K.; MITRA, G. Mean-risk models using two risk measures: a multi-objective approach. **Quantitative Finance**, Taylor & Francis, v. 7, n. 4, p. 443–458, 2007.

SARENI, B.; KRAHENBUHL, L. Fitness sharing and niching methods revisited. **IEEE transactions on Evolutionary Computation**, IEEE, v. 2, n. 3, p. 97–106, 1998.

SCHAERF, A. Local search techniques for constrained portfolio selection problems. **Computational Economics**, Springer, v. 20, n. 3, p. 177–190, 2002.

SHARPE, W. F. Capital asset prices: A theory of market equilibrium under conditions of risk. **The journal of finance**, Wiley Online Library, v. 19, n. 3, p. 425–442, 1964.

SHI, Y.; EBERHART, R. C. Empirical study of particle swarm optimization. In: IEEE. **Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)**. [S.l.], 1999. v. 3, p. 1945–1950.

SHUKLA, A.; PANDEY, H. M.; MEHROTRA, D. Comparative review of selection techniques in genetic algorithm. In: IEEE. **2015 international conference on futuristic trends on computational analysis and knowledge management (ABLAZE)**. [S.l.], 2015. p. 515–519.

SILVER, E. A.; VICTOR, R.; VIDAL, V.; WERRA, D. de. A tutorial on heuristic methods. **European Journal of Operational Research**, Elsevier, v. 5, n. 3, p. 153–162, 1980.

SPERANZA, M. G. A heuristic algorithm for a portfolio optimization model applied to the milan stock market. **Computers & Operations Research**, Elsevier, v. 23, n. 5, p. 433–441, 1996.

THAKKAR, A.; CHAUDHARI, K. A comprehensive survey on portfolio optimization, stock price and trend prediction using particle swarm optimization. **Archives of Computational Methods in Engineering**, Springer, v. 28, n. 4, p. 2133–2164, 2021.

VERSACE, M.; BHATT, R.; HINDS, O.; SHIFFER, M. Predicting the exchange traded fund dia with a combination of genetic algorithms and neural networks. **Expert systems with applications**, Elsevier, v. 27, n. 3, p. 417–425, 2004.

WOEGINGER, G. J. Exact algorithms for np-hard problems: A survey. In: **Combinatorial optimization—eureka, you shrink!** [S.l.]: Springer, 2003. p. 185–207.

WOLSEY, L. A.; NEMHAUSER, G. L. **Integer and combinatorial optimization**. [S.l.]: John Wiley & Sons, 2014.

YIN, X.; NI, Q.; ZHAI, Y. A novel particle swarm optimization for portfolio optimization based on random population topology strategies. In: SPRINGER. **International Conference in Swarm Intelligence**. [S.l.], 2015. p. 164–175.