UMA NOVA HEURÍSTICA DE SEGREGAÇÃO DE CARDUMES PARA OTIMIZAÇÃO MULTI-SOLUÇÃO DE PROBLEMAS MULTIMODAIS

Trabalho de Conclusão de Curso

Engenharia de Computação

Aluno: Marcelo Gomes Pereira de Lacerda

Orientador: Prof. Dr. Fernando Buarque de Lima Neto

UNIVERSIDADE
DE PERNAMBUCO

Marcelo Gomes Pereira de Lacerda

Uma Nova Heurística de Segregação de Cardumes Para Otimização Multi-solução de Problemas Multimodais

Monografia apresentada para obtenção do Grau de Bacharel em Engenharia de Computação pela Universidade de Pernambuco

Orientador:
Prof. Dr. Fernando Buarque de Lima Neto

GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO
ESCOLA POLITÉCNICA DE PERNAMBUCO
UNIVERSIDADE DE PERNAMBUCO

Recife - PE, Brasil

Junho de 2012

Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação
Projeto de Final de Curso

ESCOLA POLITÉCNICA DE PERNAMBUCO

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 18 de 6 de 2012, às 11:00 horas, reuniu-se para deliberar a defesa da monografia de conclusão de curso do discente MARCELO GOMES PEREIRA DE LACERDA, orientado pelo professor Fernando Buarque de Lima Neto, sob título UMA NOVA HEURÍSTICA DE SEGREGAÇÃO DE CARDUMES PARA OTIMIZAÇÃO MULTI-SOLUÇÃO DE PROBLEMAS MULTIMODAIS, a banca composta pelos professores:

Carmelo José Albanez Bastos Filho

Fernando Buarque de Lima Neto

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

☒ Aprovada ☐ Aprovada com Restrições* ☐ Reprovada

e foi-lhe atribuída nota: 9,5 (nove e meio)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O discente terá 7 dias para entrega da versão final da monografia a contar da data deste documento.

CARMELO JOSÉ ALBANEZ BASTOS FILHO

FERNANDO BUARQUE DE LIMA NETO

* Este documento deverá ser encadernado juntamente com a monografia em versão final.

Resumo

Este trabalho apresenta uma nova heurística de segregação de cardumes aplicada ao algoritmo de busca *Fish School Search* (FSS), objetivando o embasamento para a criação de um novo método de otimização multi-solução para problemas multimodais. Nesta nova abordagem, o peso dos peixes é utilizado como elemento de segregação populacional, permitindo que peixes mais pesados (i.e de mais sucesso) executem movimentos mais independentes do que os mais leves, estes últimos sendo guiados pelos primeiros. Os resultados obtidos nos experimentos mostraram que esta nova abordagem é capaz de encontrar um bom número de soluções no espaço de busca, superando as três técnicas utilizadas para comparação em 6 das 7 funções teste. Além disso, foi visto que tal abordagem requer menos esforço computacional para obter excelentes resultados. No entanto, os subcardumes formados não possuem uma alta capacidade de convergência para as soluções corretas, retornando muitas diferentes soluções como resposta. Porém, mesmo com estas limitações, pode-se concluir que o uso do peso dos peixes como operador de segregação populacional é uma ótima alternativa para ser considerada no novo algoritmo de otimização multi-solução de problemas multimodais baseado no FSS, levando em conta que as atuais limitações podem ser contornadas com algum pós-processamento.

Palavras-chave: Busca Heurística, Otimização Multi-Solução, Problema Multimodal, Fish School Search

Abstract

This work presents a new heuristic for fish school segregation applied on the Fish School Search algorithm (FSS), aiming to construct a basis for the creation of a new multi-solution optimization method for multimodal problems. In this new approach, the weight of the fishes is used as a population segregation element, allowing the haviest fishes (i.e the most succesful) to move themselves more independently and the lightest to be guided by the haviest ones. The obtained results showed that this new aproach is able to find a good amount of solutions in the search space, overcoming the three techniques used for comparison in 6 of 7 benchmark functions. Moreover, it can be seen that the new approach requires less computational effort to obtain excelent results. However, the algorithm does not have a good ability of convergence of subswarms in the right solutions, which causes the return of many different solutions. Even with such disadvantage, it can be concluded that the use of the weight of th fishes as an population segregation operator is a good alternative to be used in the new multi-solution algorithm for multimodal problem based on FSS, taking in account that the detected problems with this technique can be solved with some post-processing.

Keywords: Heuristic Search, Multi-Solution Optimization, Multimodal Problem, Fish School Search

Dedicatória

Dedico este trabalho a todos os que abdicam de momentos de descanso e diversão em nome da eterna construção da ciência, objetivando a construção de um mundo cada vez melhor.

Agradecimentos

Primeiramente, eu gostaria de agradecer a Deus por ter permitido que eu tenha chegado onde cheguei pois, sem a permissão Dele, eu nem existiria.

Eu gostaria de agradecer ao meu orientador, Prof. Fernando Buarque, por ter me introduzido à ciência e me ensinado a ser um verdadeiro cientista. Procurando sempre fazer o meu melhor, espero um dia alcançar a plenitude de tal atividade. Eu gostaria, também, de agradecer a todos os outros professores que passaram por minha vida e contribuiram com o meu crescimento, tanto pessoal quanto profissional. Gostaria, por fim, de agradecer aos funcionários da POLI/UPE que estão sempre trabalhando para fazer esta Faculdade de Engenharia um bom lugar para trabalhar, em especial, nossa "mãe" Ana Georgina.

Eu gostaria de agradecer a todos os meus colegas com os quais convivi durante estes 5 anos de muito trabalho e noites mal dormidas. Porém, eu gostaria de agradecer, em especial, aos meus amigos Paulo "Gordinho" Roger, Caio "Porquinho Indefeso" Bernardes, Erick "Elick" Barboza, Diego "Dieguito" Pinheiro, Denis "Maguinho" Martins, Rafael "Rafinha" Cabral, Bruno "Beça" Lins, Henrique "Spepe" Specht e David "Dave" Alain, por toda a amizade e companheirismo. Este trabalho tem o "dedo" de todos.

Eu gostaria de agradecer, também, a todos os meus amigos fora da faculdade, por ter me feito perceber que existe vida além da academia.

Eu gostaria de agradecer aos meus pais, Jorge e Josélia, por todo o suporte dado durante toda a minha vida, o que possibilitou todas as conquistas que obtive até hoje. Talvez, se não tivesse havido todo o estímulo pelo estudo que me foi dado, eu não estaria escrevendo, hoje, este documento. Eu gostaria, também, de agradecer ao meu irmão, Bruno, por todo seu companheirismo durante todos estes anos, me proporcionando muitos momentos inesquecíveis. Gostaria também de agradecer aos meus avós e todos os meus tios, tias, primos e primas, que, de alguma forma, contribuiram para a minha escalada, em especial aos que já não estão mais neste plano.

Por fim, eu gostaria de reservar um agradecimento especial a uma pessoa que esteve, desde o início, ao meu lado, me dando todo o suporte necessário para que

eu tenha conquistado o que hoje conquistei. Gostaria de agradecer pelo seu companheirismo, sua amizade, suas “chacoalhadas”, sua paciência, sua compreensão e todo o seu amor dado durante esse 5 anos de muito aprendizado. Devo tudo isso à minha noiva, futura esposa e eterna namorada, Caroline Lucena. Sem sua presença em minha vida, eu não teria conquistado o que conquistei.

Obrigado a todos. Este trabalho também é de vocês!

"É somente nas misteriosas equações do amor
que qualquer lógica ou razão pode ser encontrada,
você é a razão de eu estar aqui hoje,
você é a razão de eu existir,
você é todas as minhas razões."

John Nash (Interpretado por Russel Crowe no filme Uma Mente Brilhante (2001))

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas e Siglas

ABC – *Artificial Bee Colony*
ABFA – *Adaptive Bacterial Foragn Optimization*
ACO – *Ant Colony Optimization*
BFA – *Bacterial Foraging Algorithm*
BFO-PSO – *Bacterial Foraging Optimization - Particle Swarm Optimization*
dFSS – *Density based Fish School Search*
FSS – *Fish School Search*
GSO – *Glowworm Swarm Optimization*
NichePSO – *Niche Particle Swarm Optimization*
PSO – *Particle Swarm Optimization*
QS-BFO – *Quorum Sensing based Bacterial Foraging Optimization*
TSP – *Travelling Salesman Problem*

1 Introdução

Neste trabalho de conclusão de curso, uma nova heurística para segregação de cardumes (i.e metaheurística de algoritmos de enxames) é proposta com o objetivo de embasar a construção de um novo algoritmo de otimização multi-solução de funções multimodais baseado no algoritmo de busca *Fish School Search*, o qual foi inspirado no comportamento coletivo de cardumes. Neste trabalho, o peso de peixe será utilizado como elemento de segregação populacional, com o intuito de formar subcardumes exploradores de diferentes nichos, os quais representam potenciais soluções do problema.

A Seção 1.1 apresenta a motivação da realização desta pesquisa, bem como a caracterização do problema abordado. Em seguida, na Seção 1.2, são apresentadas as hipóteses de solução do problema e os objetivos deste trabalho. Por fim, a Seção 1.3 escreve a organização deste documento.

1.1 Motivação e Caracterização do Problema

Tarefas de otimização estão presentes em diversas situações vividas diuturnamente. Como exemplos, diretores de empresas precisam tomar determinadas decisões com o objetivo de maximizar o lucro de suas companhias, pilotos de automóveis em competições automobilísticas utilizam os recursos de suas máquinas de forma que estas sejam menos desgastadas mas obtenham um bom desempenho, entre muitos outros. O homem está quase sempre procurando agir de forma que obtenha o melhor resultado possível, independente de como este seja.

Define-se otimizar por ajustar um sistema com o objetivo de obter a melhor saída possível. Porém, em alguns casos, obter um “bom” resultado, não necessariamente o melhor, é suficiente [1]. Problemas de otimização possuem os seguintes elementos:

- Uma função objetivo, sendo este o alvo da otimização (minimizado ou maximizado).
- Um conjunto de variáveis, o qual influencia diretamente o valor de retorno da função objetivo.
- Um conjunto de restrições, o qual restringe os valores possíveis das dimensões da função objetivo.

Em outras palavras, otimização de uma função objetivo consiste em um processo de ajuste das variáveis desta, obedcendo às eventuais restrições impostas. Apesar de possuir uma simples definição, um problema de otimização pode ser bastante complexo, podendo possuir variáveis de diferentes tipos ou até mesmo objetivos conflitantes, podendo estes ter suas características alteradas com o decorrer do tempo. Tais problemas podem ser classificados de acordo com o número de variáveis (única variável ou multivariável) e seus tipos (discreto ou contínuo), o grau de não-linearidade da função objetivo (linear ou não-linear), a ocorrência ou não de alterações em suas características com o decorrer do tempo (estático ou dinâmico), as restrições do problema, o número de soluções ótimas (unimodal ou multimodal) e o número de funções objetivo (único objetivo ou multiobjetivo) [2].

No mundo real, problemas de otimização multimodais são os mais comuns, sendo estes, normalmente, de difícil solução. Segundo Madeiro [3], estes são presentes em áreas como geofísica, eletromagnetismo, climatologia e logística. Por este motivo, problemas dessa classe vêm recebendo bastante atenção atualmente.

Algoritmos de Otimização são técnicas que executam buscas pela solução de um determinado problema de otimização, este representado por uma função objetivo. Diversos modelos computacionais inspirados na natureza e voltados para a resolução de problemas de otimização vêm sendo criados e explorados. Sendo o foco deste trabalho, Inteligência de Enxames é definida como a propriedade de um sistema no qual a interação entre uma população de indivíduos de comportamento simples gera padrões funcionais complexos [4]. Como exemplos de algoritmos que possuem tal comportamento pode-se citar a Otimização por Enxame de Partículas, ou *Particle Swarm Optimization* (PSO), o qual consiste em um algoritmo baseado no comportamento coletivo de pássaros [5], o ACO (*Ant Colony Optimization*), meta-heurística inspirada na busca por alimentos executada por colônias de formigas [6], e o FSS (*Fish School Search*), algoritmo inspirado no comportamento de cardumes de peixes [7]. No algoritmo

FSS, os indivíduos (peixes) possuem seu mecanismo de movimentação dividido em três fases: movimento individual, movimento coletivo instintivo e movimento coletivo volitivo. O sucesso da busca é representado através do peso de cada peixe, o qual aumenta sempre que este melhora o seu *fitness*, ou seja, sempre que este passa a representar uma solução melhor do que a solução representada na iteração anterior. Desta forma, o peso influencia diretamente na direção dos movimentos coletivos [7].

Para problemas de otimização multi-solução, podem ser citadas técnicas como CPSO (*Craziness* PSO) [8], Multi_PSOer (Multi_Optimizer PSO) [9], NichePSO [10] e GSO (*Glowworm Swarm Optimization*), um algoritmo baseado no comportamento social de vagalumes [11].

Uma versão multimodal do FSS foi proposta por Madeiro [3][12], denominada de dFSS (*Density based Fish School Search*), a qual utiliza o conceito de densidade como fator de segregação populacional. Foi introduzido no algoritmo original mecanismos de compartilhamento de alimentos, além de ter sido inserido uma memória em cada indivíduo, a qual armazena o histórico de influência que cada peixe exerceu no detentor da mesma. Em todas as funções de teste utilizadas, o dFSS obteve melhores resultados em comparação com o GSO e o NichePSO.

Como afirmado anteriormente, este trabalho visa a realização de experimentos nos quais o peso do peixe é utilizado como fator de segregação populacional. Os resultados aqui apresentados devem servir de base para a criação de um novo método de otimização multi-solução baseado no FSS original, este sendo uma alternativa ao dFSS.

1.2 Hipóteses e Objetivos

Detalhado na Seção 2.2.3, o algoritmo dFSS faz uso de informações espaciais para executar um operador de partilha de alimento. Através desta partilha e acrescentando memória nos peixes, o que não existia no FSS original, o cardume principal é dividido em subcardumes ao longo das iterações, os quais, facilitados por outros modificações feitas nos operadores do FSS, devem concentrar-se em nichos com vista a explorá-los.

Neste trabalho, são realizados experimentos com o uso do peso do indivíduo como elemento de segregação populacional. A divisão do cardume em subcardumes é feita

através da definição de relações do tipo *guia-guiado*, onde os peixes mais pesados tendem a guiar os mais leves, pois quanto mais pesado um peixe for, melhor foi o seu processo de busca até o momento, sendo, portanto, mais razoável o encaminhamento de peixes mais leves (pelos mais pesados) para bons lugares. O peso do peixe é considerado como a representação de sucesso do algoritmo FSS. Portanto, fazer uso do mesmo como critério de definição de guias e, conseqüentemente, resultar na segregação do cardume principal em subcardumes, pode levar a uma processo de divisão e exploração de nichos de forma mais natural se comparado ao uso de informações espaciais como a distância euclidiana e o conceito de vizinhança. Além disso, o peso do peixe é um valor já computado e atualzado no algoritmo, o que descarta a necessidade de executar computações extras, o que aconteceria com o uso, por exemplo, de distância euclidiana.

1.2.1 Objetivos Gerais

- Analisar os efeitos da utilização do peso do peixe como mecanismo de segregação em problemas multimodais.

1.2.2 Objetivos Específicos

- Propor um mecanismo de uso do peso do peixe nos cálculos de sua movimentação individual e coletiva, visando a segregação do cardume;
- Implementar os mecanismos propostos;
- Planejar e executar os experimentos;
- Analisar os resultados obtidos.

1.3 Organização do Documento

Este trabalho está dividido em 6 capítulos. No corrente capítulo, uma introdução ao problema foi feita. No Capítulo 2, é apresentada uma revisão bibliográfica sobre o assunto. No Capítulo 3, é apresentada a abordagem proposta neste trabalho. Em seguida, no Capítulo 4, os experimentos utilizados para validar a abordagem são descritos. No Capítulo 5, os resultados dos experimentos são apresentados. Por fim, no

Capítulo 6, são apresentadas as conclusões acerca deste trabalho, além de planos para trabalhos futuros. No final deste documento, está disponível um Apêndice com os gráficos referentes aos resultados dos experimentos apresentados no Capítulo 5 caso o leitor deseje realizar uma consulta mais aprofundada acerca dos mesmos.

2 Fundamentação Teórica

Neste capítulo, são introduzidos os conceitos necessários para um melhor entendimento deste trabalho. Na Seção 2.1, são explicados conceitos gerais sobre problemas de otimização, abordando os elementos básicos, tipos de ótimos existentes e tipos de problemas de otimização. Em seguida, na Seção 2.2, conceitos acerca de métodos de otimização existentes são introduzidos, além de serem abordados tópicos relacionados a Inteligência de Enxames, citando exemplos de técnicas pertencentes a esta área, entrando em mais detalhes ao citar o algoritmo *Fish School Search*, este último por servir de base para a proposta deste trabalho. São também abordados em mais datalhes alguns métodos de otimização multi-solução (*Niche Particle Swarm Optimization*, *Glowworm Swarm Optimization*, e *Density based Fish School Search*), sendo este tipo de técnica o alvo deste trabalho.

2.1 Problemas de Otimização

Segundo Kennedy *et al.* [1], existem algumas divergências acerca da definição do termo “otimização”. Otimizar pode ser entendido como ajustar os parâmetros de um sistema objetivando a obtenção do melhor retorno possível. Porém, como citado no capítulo anterior, pode-se também ser entendido como um ajuste patamétrico do sistema, de forma que este retorne uma “boa” saída, não necessariamente a melhor possível, pois, em muitos casos, encontrar a melhor solução de um problema pode ser inviável. Neste trabalho, será utilizado o segundo conceito citado como definição de problemas de otimização.

2.1.1 Elementos Básicos

Problemas de otimização possuem os seguintes elementos:

- Uma função objetivo, o que representa, matematicamente, o problema a ser minimizado ou maximizado. Como exemplo, pode-se citar $f(x) = x^2 + x + 1$, como um problema com uma única variável, ou $f(x,y,z) = x^2 + 3y^4 + z$, como um problema com três variáveis. Porém, muitos problemas do mundo real não possuem uma função objetivo explicitamente definida. Nestes casos, o processo de busca deve retornar soluções que satisfaçam o conjunto de restrições inerente aos mesmos.

- Um conjunto de variáveis, o qual define o valor de retorno da função objetivo. Por exemplo, em $f(x,y,z) = x^2 + 3y^4 + z$, o conjunto $V = \{x,y,z\}$ representa o conjunto de variáveis de f.

- Um conjunto de restrições, que restringe os valores possíveis a serem atribuídos às variáveis do problema. As restrições podem apenas limitar os valores das variáveis em máximo e mínimo ou até mesmo excluir algumas potenciais soluções do conjunto de soluções possíveis. Como exemplo de uma restrição, pode-se determinar que as variáveis de um determinado problema receberão apenas valores positivos, ou seja, terão seus valores, invariavelmente, contidos no intervalo $[0, +\infty)$ [2].

2.1.2 Tipos de Ótimos

Soluções de problemas de otimização podem ser classificadas em dois grandes grupos: ótimos locais ou ótimos globais [2]. Porém, estas podem ser detalhadamente classificadas de acordo com as seguintes definições formais (considerando problemas de minimização):

- Mínimo Global Forte ou Estrito: Uma solução $x^* \in F$ é considerada um mínimo global forte ou estrito de uma função f se $f(x^*) < f(x), \forall x \in F$, onde $F \subseteq S$, sendo S o conjunto de soluções possíveis. Dessa forma, f possui apenas um mínimo global [2][3].

- Mínimo Global Fraco: Uma solução $x^* \in F$ é considerada um mínimo global fraco de uma função f se $f(x^*) \leq f(x), \forall x \in F$, onde $F \subseteq S$, sendo S o conjunto de soluções possíveis. Dessa forma, f possui mais de um mínimo global [3].

- Mínimo Local Forte ou Estrito: Uma solução $x^*_N \in N$ é considerada um mínimo local forte ou estrito de uma função f se $f(x^*_N) < f(x), \forall x \in N$, onde $N \subseteq F$ e N é um

conjunto dos pontos mais próximos a x_N^*, sendo $F \subseteq S$ e S o conjunto de soluções possíveis. Dessa forma, x_N^* é a única solução existente na sua vizinhança [2][3].

- Mínimo Local Fraco: Uma solução $x_N^* \in N$ é considerada um mínimo local fraco de uma função f se $f(x_N^*) \leq f(x), \forall x \in N$, onde $N \subseteq F$ e N é um conjunto dos pontos mais próximos a x_N^*, sendo $F \subseteq S$ e S o conjunto de soluções possíveis. Dessa forma, existe mais de uma solução na vizinhança de x_N^* [2][3].

2.1.3 Tipos de Problemas de Otimização

Segundo Engelbrecht [2], problemas de otimização pode ser classificados de acordo com as seguintes características:

- Número de variáveis;
- Tipos das variáveis;
- Grau de não-linearidade da função objetivo;
- Restrições utilizadas;
- Número de soluções ótimas;
- Número de critérios de otimização.

Algumas destas classificações foram descritas no Capítulo 1. As demais serão detalhadas em seguida.

Grau de não-linearidade da função objetivo

O problema pode ser classificado como linear ou não-linear dependendo do grau de linearidade da função objetivo. Por exemplo, a função $f(x) = x + 3$ representa um problema linear, enquanto $f(x) = x^3 + 5x^2 + 1$ representa um problema não-linear.

Restrições utilizadas

Problemas que usam apenas limites máximo e mínimo dos valores das variáveis são considerados problemas sem restrições. Para que um problema possua restrições, neste devem estar definidas inequalidades que excluam algumas soluções do

espaço de soluções possíveis, o que é comum em problemas do mundo real, pois algumas soluções são impossíveis de serem alcançadas.

Formalmente, considerando apenas problemas de minimização, problemas sem restrições e com restrições são definidos pelas Equações (2.1) e 2.2, respectivamente. Na Equação (2.1), $x \in F = S$ e $dom(x_j)$ é o domínio de x_j ($\mathbb{R}$ se for contínuo e $\mathbb{Z}$ se for discreto, por exemplo). Na Equação (2.2) n_g e n_h são os números de equalidades e inequalidades, respectivamente, e $dom(x_j)$ é o domínio de x_j.

$$\begin{aligned} &minimizar \;\; f(x), x = (x_1, x_2, ..., x_n), \\ &x_j \in dom(x_j). \end{aligned} \tag{2.1}$$

$$\begin{aligned} &minimizar \;\; f(x), x = (x_1, x_2, ..., x_n), \\ &g_m(x) \leq 0, m = 1, ..., n_g, \\ &h_m(x) = 0, m = n_g + 1, ..., n_g + n_h, \\ &x_j \in dom(x_j). \end{aligned} \tag{2.2}$$

Dinamicidade da função objetivo

Alguns problemas possuem uma função objetivo que muda com o decorrer do tempo. Tais problemas são denominados dinâmicos, enquanto os que não possuem suas características alteradas são chamados de estáticos. É importante observar que, devido a tais mudanças nas características da função objetivo, o(s) ótimos podem ser deslocados, além desaparecerem ou até surgirem novos.

Formalmente, considerando apenas problemas de minimização, problemas dinâmicos são definonde, n_g e n_h são os números de equalidade e inequalidade, respectivamente, e $dom(x_j)$ é o domínio de x_j.idos pela Equação (2.3).

$$\begin{aligned} &minimizar \;\; d(x, \omega(t)), x = (x_1, ..., x_n), \omega(t) = (\omega_1(t), ..., \omega_n(t)) \\ &g_m(x) \leq 0, m = 1, ..., n_g \\ &h_m(x) = 0, m = n_g + 1, ..., n_g + n_h \\ &x_j \in dom(x_j) \end{aligned} \tag{2.3}$$

onde, $\omega_i(t)$ é uma função objetivo variante no tempo determinante do valor da variável x_i no instante t, n_g e n_h são os números das equalidades e inequalidades restritivas (caso existam), respectivamente, e $dom(x_j)$ é o domínio de x_j. Em outras palavras, o processo de otimização de um problema dinâmico deve localizar o(s) ótimo(s) a cada instante de tempo, ou seja, se algum ótimo muda, o algoritmo deve mantê-lo rastreado.

Número de soluções ótimas

Problemas unimodais são possuidores de apenas uma solução. Problemas com mais de uma solução são denominados multimodais. Como será descrito formalmente mais adiante, problemas multimodais podem ter todos os seus ótimos sendo globais ou alguns destes podem ser locais enquanto outros são globais.

Formalmente, considerando apenas problemas de minimização, problemas unimodais e multimodais são definidos pelas Equações (2.4) e (2.5), respectivamente.

$$f(x^*) \leq f(x), \forall x \in \mathbb{R}^n, \tag{2.4}$$

$$f(x_L^*) \leq f(x), \forall x \in L, L \subset \mathbb{R}^n, \tag{2.5}$$

onde x^* é o único ótimo global, x_L^* é um dos ótimos locais, $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ é a função objetivo e n é o número de dimensões do espaço de busca.

2.2 Métodos Meta-Heurísticos de Otimização

Método de otimização é um algoritmo cujo objetivo é encontrar uma boa solução para um determinado problema, transformando, durante sua execução, soluções candidatas com alguma qualidade associada em uma solução, na maioria dos casos, melhor [2][13]. Tais técnicas podem ser divididas em dois grupos: algoritmos de busca local e algoritmos de busca global. Os algoritmos de busca local utilizam apenas informações da sua vizinhança para realizar sua tarefa, o que resulta em uma localização de ótimos locais. Os algoritmos de busca global fazem uso de outros recursos, como, por exemplo, comunicação entre indivíduos distribuídos em todo o espaço de busca, para realizar uma busca em toda a superfície do espaço de soluções possíveis, obje-

tivando encontrar o ótimo global do problema [2].

Métodos de otimização podem ser também classificados entre métodos determinísticos e métodos estocásticos [2][13]. Entre os métodos determinísticos, existem os métodos exatos finitos e os métodos heurísticos. Os métodos exatos finitos garantem que o ótimo global seja encontrado em um tempo finito. No entanto, este tempo pode ser relativamente alto. Já os métodos heurísticos não garantem que o ótimo global seja encontrado, porém garante a busca por uma "boa" solução em um tempo hábil, fazendo uso de conhecimentos prévios que direcionam os passos a serem tomado durante a busca, excluindo a necessidade de iteração em todas as soluções [13][14].

Métodos estocásticos fazem uso de operadores aleatórios, o que não permite uma previsão exata de como será o próximo estado do processo de busca, dado o estado atual do mesmo [2][15]. Assim como no grupo dos métodos determinísticos, existem métodos estocásticos heurísticos, sendo este o foco deste trabalho.

Após esta revisão sobre problemas e métodos de otimização, na próxima seção, serão abordados conceitos referentes à Inteligência de Enxames, onde serão detalhados alguns algoritmos meta-heurísticos, sendo estes necessários para o entendimento da abordagem proposta neste trabalho.

2.2.1 Inteligência de Enxames

Diversos modelos computacionais inspirados na natureza e voltados para a resolução de problemas de otimização vêm sendo criados e explorados. Como citado no capítulo anterior, Inteligência de Enxames é definida como a propriedade de um sistema no qual a interação entre indivíduos simples de uma população gera padrões funcionais complexos, permitindo que problemas de alta complexidade sejam resolvidos sem a necessidade de um controle central, de forma que tal controle esteja distribuído nas entidades [4].

Inicialmente introduzidos em revistas e conferências voltadas para discussões sobre Computação Evolucionária [16], os métodos de Inteligência de Enxames vêm ganhando uma atenção cada vez maior da comunidade acadêmica, sendo cada vez mais aplicados a problemas de otimização [17]. Basicamente, o funcionamento de tais técnicas baseia-se na transmissão de informações adquiridas localmente por um determinado indivíduo para outros indivíduos do enxame. Dessa forma, uma rede de conhecimento é criada, onde a circulação de informações locais faz emergir um

conhecimento global acerca do espaço de busca. Fazendo uso dessas informações, as entidades comportam-se de forma que suas movimentações as levem para locais onde, provavelmente, encontrarão as melhores soluções possíveis. Tais comportamentos são modelados inspirados em entidades presentes na natureza [3][17]. Segue abaixo uma lista com resumos de diferentes técnicas utilizando metáforas com características distintas:

- ***Particle Swarm Optimization (PSO)*** - Em 1995, Kennedy e Eberhart [5][18] propuseram uma técnica de otimização de funções fazendo uso de uma busca heurística inspirada no comportamento coletivo de bandos de pássaros. O propósito inicial era simular graficamente o comportamento social de tais bandos, tentando encontrar, desta forma, padrões que permitissem que os indivíduos voassem em uma determinada direção e, de forma sincronizada, mudassem de direção, mas sem perder a formação. Porém, como resultado desta inverstigação, um simples e poderoso algoritmo de otimização foi criado. A movimentação de cada partícula (pássaro) é guiada por duas componentes: componente cognitiva e componente social. A componente cognitiva caracteriza-se pela memória pessoal de cada indivíduo sobre seu processo de busca, a qual guarda a melhor posição já encontrada, enquanto a componente social armazena a melhor posição encontrada por um grupo de partículas, a qual o indivíduo em questão faz parte. Dessa forma, o processo de "decisão" da direção a ser tomada é definido por uma composição ponderada entre tais componentes, fazendo com que as partículas possuam a tendência ao movimento nda direção das boas regiões do espaço de busca. Poucos anos após a primeira publicação científica do algoritmo, Shi, Eberhart [19] Clerc e Kennedy [20] propuseram a adição de novos componentes no cálculo da atualização da velocidade das partículas, aumentando ou garantindo a convergência do algoritmo. Uma extensa revisão acerca do algoritmo PSO foi escrita por Banks *et al.*, a qual foi dividida em dois artigos. [21][22].

- ***Ant Colony Optimization (ACO)*** - Proposto por Dorigo em sua tese de doutorado em 1992 [6] e concebido para resolver problemas combinatoriais, como, por exemplo, o clássico *Travelling Salesman Problem* (TSP) [23], o algoritmo *Ant Colony Optimization* possui seu mecanismo baseado no comportamento coletivo forrageiro de colônias de formigas. Tal algoritmo é comumente utilizado em problemas modelados por grafos. Igualmente ao comportamento explora-

tório das formigas naturais à procura de rotas que as levem a boas fontes de alimentos, maximizando a quantidade de alimento encontrado e minimizando a complexidade do caminho percorrido, as formigas artificiais buscam melhores caminhos em grafos, de forma que tal percurso represente uma solução válida ao problema. No mundo real, de acordo com a qualidade da fonte encontrada, as formigas depositam uma determinada quantidade de ferormonio no caminho percorrido, sendo esta quantidade proporcional à tal qualidade. No algoritmo, o depósito proporcional à qualidade da solução é representado pela modificação do peso das arestas do grafo. Tal depósito caracteriza a comunicação entre o indivíduos, o qual, diferente do PSO, onde a partículas comunicam diretamente seus sucessos durante seus processos de busca, esta é feita indiretamente através da modificação do ambiente. Tal tipo de colaboração é chamada de estigmergia [24]. A movimentação de cada formiga é guiada pela concentração de ferormonio em cada aresta do grafo, sendo a probabilidade de escolha de um determinado caminho diretamente proporcional à quantidade de ferormonio depositado neste. Como conseqüência deste mecanismo, as formigas tendem a seguir os caminhos mais bem avaliados pela população durante todo o processo de busca. A primeira modificação com melhorias significativas foi proposta por Dorigo e Gambardella [23], na qual, entre outras modificações menores, acrescenta-se uma atualização local do ferormonio depositado nas arestas, a qual é feita logo após o percorrimento de cada aresta.

- ***Artificial Bee Colony (ABC)*** - Proposto por Karaboga [25][26], o algoritmo foi desenvolvido tendo como inspiração o comportamento coletivo em colméias de abelhas, no qual as abelhas cooperam entre si no processo de busca por fontes de alimentos em prol do desenvolvimento e manutenção da colméia. Assim como nos enxames naturais, as abelhas artificiais são dotadas de especialização, ou seja, existe mais de um tipo de abelha, sendo cada tipo responsável por uma operação específica. No algoritmo ABC, algumas abelhas devem explorar o espaço de busca de forma independente à procura de boas fontes de alimentos. Encontradas tais fontes, outras abelha ficam encarregadas apenas de executar uma exploração mais refinada nas proximidades de cada fonte. Para tal, cada abelha deve escolher qual fonte irá explorar durante um determinado período, sendo a probabilidade de escolha proporcional à qualidade da mesma. Tal exploração refinada distribuída em diferentes focos permite um bom desempenho do algoritmo em problema multimodais. Algumas modificações do algoritmo fo-

ram propostas, como, por exemplo, uma versão do algoritmo desenvolvida para a resolução de problemas com restrições [27], além de uma versão paralela do mesmo [28].

- ***Bacterial Foraging Algorithm (BFA)*** - O BFA [29] é um algoritmo de otimização baseado no comportamento forrageiro das bactérias, especificamente da espécie E. Coli. Nesta abordagem, os indivíduos (bactérias) movem-se sempre para locais com uma maior concentração de nutrientes. Em outras palavras, as bactérias movem-se sempre em direção a regiões que aumentem os seus respectivos *fitnesses*. Existem três operadores no BFA: Operador de Movimento Quimiotáxico, Operador Eliminativo-Dispersivo e Operador de Reprodução. O Operador de Movimento Quimiotáxico é responsável, em cada bactéria, pelas buscas das regiões com maior concentração de nutrientes. Neste operador também ocorre a comunicação entre os indivíduos, a qual é feita indiretamente (assim como ocorre no ACO) através de alterações realizadas na superfície do espaço de busca, fazendo uso de elementos denominados de atratores e repulsores. Estes devem informar possíveis boas regiões a bactérias que estão a uma determinada distância do indivíduo em questão. O Operador de Reprodução é responsável pela convergência do algoritmo. Nesta fase, os $\frac{S}{2}$ melhor indivíduos são duplicados, dando origem a indivíduos idênticos e localizados na mesma posição no espaço de busca, onde S corresponde ao número total de indivíduos presentes na população. A outra metade é removida da colônia. O Operador Eliminativo-Dispersivo é responsável pela manutenção da diversidade. Este operador, de acordo com uma probabilidade definida previamente pelo usuário, faz com que algumas bactérias sejam escolhidas através de uma seleção aleatória para serem reposicionadas em outro local definido aleatoriamente. Para que não ocorra convergência prematura do algoritmo ou a busca não se torne um processo predominantemente aleatório, tal mecanismo deve estar em equilíbrio com o processo de reprodução. Algumas modificações foram propostas, como, por exemplo, uma hibridização entre o algoritmo PSO e o BFA (BFO-PSO) [30], no qual as bactérias podem mover-se como uma partícula no PSO ou uma bactéria no BFA original, um BFA com o processo de movimento quimiotáxico adaptativo (ABFA - *Adaptive Bacterial Foraging Algorithm*) [31] e um algoritmo no qual um mecanismo natural denominado de *Quorum Sensing* é utilizado no algoritmo (QS-BFO - *Quorum Sensing based Bacterial Foraging Optimization*) [32], no qual as bactérias são capazes de “contar” quantos indivíduos estão próximos a elas e, assim,

modificarem seus comportamentos.

2.2.2 *Fish School Search*

O algoritmo *Fish School Search*, proposto por Bastos Filho e Lima Neto em 2009 [7], é inspirado no comportamento coletivo de um cardume de peixes. Na natureza, muitas espécies vivem em bandos, objetivando aumentar suas chances de sobrevivência. Em cardumes de peixes, tais indivíduos podem funcionar, coletivamente, como um único organismo, objetivando, por exemplo, a proteção mútua em relação aos predadores e a busca conjunta por alimentos.

Alguns comportamentos serviram de inspiração natural para a construção passada e futura do algoritmo FSS:

- Alimentação: Naturalmente, indivíduos alimentam-se por questão de sobrevivência, objetivando o crescimento saudável e a habilitação para a reprodução. Neste algoritmo, a quantidade de alimento ingerido é utilizada como critério de avaliação de qualidade das soluções, as quais são representadas pelos peixes;
- Nado: O processo de nado, o qual ocorre de forma coordenada, permite que o cardume realize buscas por alimento em seu espaço de busca. Dessa forma, este é guiado pela necessidade de alimentação;
- Reprodução: A ocorrência de reprodução significa que o processo de busca vem sendo bem sucedido. Este possui o objetivo de executar uma busca mais refinada em bons locais encontrados pelo cardume, permitindo a sobrevivência dos mais adaptados e a eliminação dos menos sucedidos.

Apesar do comportamento de reprodução estar presente no primeiro trabalho onde o FSS foi proposto, este comportamento não foi utilizado na versão inicial do algoritmo. O FSS é descrito, sucintamente, pelo Algoritmo 1. Neste algoritmo, o sucesso é representado pelo peso de cada indivíduo. Ou seja, quanto mais pesado um peixe for, maior sucesso este obteve em seu histórico do processo de busca. Tal massa é adquirida através do processo de alimentação. Uma outra indicação de sucesso, esta coletiva, é a diminuição do raio do cardume, como será explicado a seguir.

Em seguida, todos os operadores do FSS serão descritos.

Algoritmo 1: Algoritmo do FSS.

```
P:População de peixes;
enquanto Condição de parada não é satisfeita faça
    para cada peixe em P faça
        Executa o Movimento Individual;
    para cada peixe em P faça
        Executa o processo de alimentação;
    para cada peixe em P faça
        Executa o Movimento Coletivo Instintivo;
    Calcula o baricentro do cardume;
    para cada peixe em P faça
        Executa o Movimento Coletivo Volitivo;
    Retorna a melhor soluçaõ encontrada;
```

Operador de Movimento Individual No Operador de Movimento Individual, cada indivíduo executa um movimento aleatório e independente, porém, sempre na direção do gradiente positivo, se for o problema for de maximização, ou negativo, se for um problema de minimização. Em outras palavras, este operador é executado apenas se a nova posição for melhor do que a anterior. Tal movimento é descrito pela Equação (2.6), onde $x_{ij}(t)$ é o valor da dimensão j do vetor de posição do peixe i no tempo t, $x_{ij}(t+1)$ é o seu novo valor, r é uma variável aleatória pertencente ao intervalo $[-1,1]$ e $step_{ind}$ é o tamanho do passo individual no tempo t. Objetivando a obtenção de uma busca mais ampla no início da execução do algoritmo e um refinamento gradual da mesma com o passar das iterações, o tamanho do passo individual pode decair linearmente, como descreve a Equação (2.7), onde $step_{ind}(t)$ e $step_{ind}(t+1)$ são os tamanhos do passo do movimento individual antes e depois de sua atualização, $step_{indinicial}$ é o tamaho de tal passo inicial, $step_{ind\,final}$ é o seu tamanho final e $iterations$ é o número máximo de iterações do algoritmo.

$$x_{ij}(t+1) = x_{ij}(t) + r \;\; step_{ind}(t), \tag{2.6}$$

$$step_{ind}(t+1) = step_{ind}(t) - \frac{step_{ind_{init}} - step_{ind_{final}}}{iterations}. \tag{2.7}$$

Operador de Alimentação O Operador de Alimentação é responsável por atualizar o peso do peixe de acordo com a quantidade de alimento ingerido através do Operador de Movimento Individual, ou seja, de acordo com o valor de melhora do *fitness* após o

tal movimento. Esta atualização é descrita pela Equação (2.8), onde $W_i(t)$ e $W_i(t+1)$ são os valores do peso do peixe i antes e depois da alimentação, respectivamente, Δf_i, Δf_j e Δf_a são as variações de *fitness* após o Movimento Individual dos peixes i, j e a, respectivamente e n é o tamanho do cardume. Como pode ser notado, a alimentação é feita de acordo com o aumento relativo do *fitness* do indivíduo, sendo o peixe com maior ganho em determinada iteração aquele que apresentar o maior aumento deste valor.

$$W_i(t+1) = W_i(t) + \frac{\Delta f_i}{\{\Delta f_j | \forall a \in [0, n-1] : \Delta f_a \leq \Delta f_j\}}. \tag{2.8}$$

Operador de Movimento Coletivo Instintivo O Operador de Movimento Coletivo Instintivo é o primeiro movimento coletivo a ser executado em cada iteração do algoritmo. É responsável por definir um vetor o qual será adicionado ao vetor de posíção de todos os indivíduos. Em outras palavras, neste operador, os indivíduos devem "imitar" o comportamento dos mais bem sucedidos. Tal operador é descrito pela Equação (2.9), onde $x_{ij}(t)$ é o valor da dimensão j do vetor de posição do peixe i no tempo t, $x_{ij}(t+1)$ é tal valor atualizado, ou seja, no tempo $t+1$, N é o número de indivíduos na população, Δx_{kj} é a variação do valor da dimensão j do vetor de posição do peixe k antes e depois do último Movimento Individual e $\Delta f(x_k)$ é variação do fitness do mesmo peixe.

$$x_{ij}(t+1) = x_{ij}(t) + \left(\frac{\sum_{k=1}^{N} \Delta x_{kj} \Delta f(x_k)}{\sum_{k=1}^{N} \Delta f(x_k)} \right). \tag{2.9}$$

Operador de Movimento Coletivo Volitivo Neste operador, o cardume deve realizar contrações ou expansões tendo como referência o baricentro do cardume, o qual é calculado utilizando o peso de cada peixe, de acordo com a Equação (2.10), onde $B_j(t)$ é o valor da dimensão j do baricentro no tempo corrente, N é o tamanho da população, x_{ij} é o valor da componente j do vetor de posição do indivíduo i e $W_i(t)$ é o peso do indivíduo i no tempo t. Em cada execução do movimento coletivo volitivo, o peso total do cardume deve ser calculado. Se tal valor aumenta, o cardume deve contrair de forma que uma busca local seja executada em uma determinada região, pois tal aumenta significa um processo de busca bem sucedido. Por outro lado, se este valor não aumenta, o cardume deve expandir, pois o local explorado não está oferecendo boas fontes de alimento. Com esta expansão, a população evita a estagnação em mínimos locais, realizando, desta forma, uma exploração mais ampla, ou seja,

buscando outras regiões. Tal processo é descrito pelas Equações (2.11) (contração) e (2.12) (expansão), onde $step_{vol}$ é o tamanho do passo volitivo definido previamente pelo usuário, o qual, assim como ocorre com o tamanho do passo individual, pode ser decrescido linearmente ao decorrer da execução do algoritmo, $rand(0,1)$ é um número aleatoriamente gerado entre 0 e 1 e $distance(x_i(t), B_j(t))$ é uma função que retorna a distância euclidiana entre os vetores $x_i(t)$ e $B_j(t)$.

$$B_j(t) = \frac{\sum_{i=1}^{N} x_{ij}(t) W_i(t)}{\sum_{i=1}^{N} W_i(t)}, \tag{2.10}$$

$$x_{ij}(t+1) = x_{ij}(t) - step_{vol} rand(0,1) \frac{(x_{ij}(t) - B_j(t))}{distance(x_i(t), B_j(t))}, \tag{2.11}$$

$$x_{ij}(t+1) = x_{ij}(t) + step_{vol} rand(0,1) \frac{(x_{ij}(t) - B_j(t))}{distance(x_i(t), B_j(t))}. \tag{2.12}$$

2.2.3 Algoritmos de Nicho

Como citado na Seção 2.1.3, problemas multimodais possuem mais de um ótimo, podendo estes serem globais ou locais. Algumas técnicas possuem a capacidade de retornar mais de uma solução ao usuário. Tais abordagen são chamadas de Métodos de Otimização Multi-Solução. Tais técnicas também são chamadas de Algoritmos de Nicho, pois as soluções são retornadas a partir da exploração de diferentes regiões do espaço de busca (fontes) por grupos de indivíduos pertencentes à população, sendo tais regiões denominadas de *nicho* [33]. Tais métodos podem ser classificados de acordo com a forma com a qual estes localizam os nichos [2]:

- **Busca seqüencial**: Várias buscas por um único ótimo são executadas seqüencialmente, encontrando um nicho em cada execução e removendo-o do espaço de busca para que outra busca seja executada e esta não retorne o nicho recém removido. Para tal remoção, uma modificação no espaço de busca é necessária.
- **Busca paralela**: Todos os nichos devem ser localizados paralelamente. Nesta abordagem, o algoritmo necessita possuir a capacidade de dividir sua população em grupos, os quais devem dirigir-se a diferentes regiões do espaço de busca e realizar buscas locais refinadas. Durante a execução do algoritmo, os diferentes grupos populacionais devem convergir para um ponto cetral a cada um, man-

tendo uma exploração ao redor de tal ponto, sendo tal região considerada um ótimo local do espaço de busca [34].

- **Quasi-sequential niching**: A busca pelos nichos ocorre seqüencialmente, mas sem alterar o espaço de busca na remoção do nicho. Após encontrar um nicho, a busca por outros continua, porém, mantendo uma exploração mais refinada nos nichos já encontrados.

Algumas técnicas de Otimização Multi-Solução são detalhadas nas próximas seções, sendo estas as técnicas que servirão de comparativo para a avaliação dovos mecanismos propostos neste trabalho.

Niche Particle Swarm Optimization

Proposto por Brits *et al.* [10], o algoritmo *Niche Particle Swarm Optimization* (NichePSO) é baseado no PSO original.

No início da execução do algoritmo, as partículas do enxame inicial, conhecido como enxame principal, estão totalmente desconectadas entre si. Após a inicialização dos indivíduos, dá-se início um processo denominado de treinamento do enxame principal. Nesta processo, as partículas se movimentam de forma independente, seguindo o mecanismo representado pelas Equações (2.13) e (2.14), o qual foi proposto por Kennedy em 1999 [35] e foi denominado de modelo “somente cognição”, onde $v_{ij}(t+1)$, $v_i(t)$, $x_{ij}(t+1)$ e $x_i(t)$ são os valores do componente j dos vetores de velocidades e posição, respectivamente, da partícula i nos tempos $t+1$ e t, respectivamente, w é denominado de fator de inércia, $c1$ é uma constante definida previamente pelo usuário, $r1$ é uma variável aleatória e $p_{best_ij}(t)$ é o valor do componente j do vetor da melhor posição já encontrada pela partícula i no espaço de busca até o momento t.

$$v_{ij}(t+1) = wv_i(t) + c1r1(t)(p_{best_ij}(t) - x_{ij}), \tag{2.13}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1). \tag{2.14}$$

A formação de subenxames é feita através do monitoramento constante do desempenho das partículas do enxame principal durante o treinamento do enxame principal.

Neste processo de monitoramento, as partículas que passarem a não mais melhorar dentro de um determinado intervalo de tempo serão consideradas como uma potencial solução do problema. A partir disso, a partícula mais próxima a esta será escolhida para, junto com o indivíduo estagnado, formar um subenxame, sendo ambos removidos do enxame principal. A verificação de estagnação é feita através do cálculo do desvio-padrão do fitness da partícula a cada n iterações, sendo n definido previamente pelo usuário. Se tal desvio-padrão for menor do que um determinado limiar também definido pelo usuário, a partícula é considerada estagnada.

As partículas pertencentes ao enxame principal não possuem conhecimentos sobre os nichos descobertos pelos subenxames. Por isso, algumas partículas podem explorar regiões já exploradas por subenxames. Para evitar tal situação, as partículas que invadirem o espaço pertencente a um determinado subenxame será acrescentado a tal grupo. Tal invasão é detectada se a partícula estiver dentro do raio de atuação do subenxame, o qual é definido pela máxima distância entre o G_{best} do subenxame (o melhor ponto já visitado por todas as partículas do subenxame em todo o histórico de busca) e uma partícula pertencente ao mesmo.

Após todas as partículas serem removidas do enxame principal, não é mais possível realizar nenhuma criação de novos subenxames. Porém, é possível fundir diferentes subenxames em um só, caso suas hiperesferas, dimensionadas pelos seus respectivos raios de atuação, se interceptam. Para a fusão dos dois subenxames, as partículas do grupo com menor G_{best} são transferidas para o grupo mais bem sucedido. Ao final da execução do algoritmo, os $G_{best}s$ de cada subenxame será retornado como as soluções encontradas pelo NichePSO.

Em 2007, Engelbrecht e van Loggerenberg [36] propuseram algumas alterações nos métodos de fundição de subenxames e absorção de partículas a subenxames já criados, as quais resultaram em melhorias significativas no desempenho do algoritmo.

No novo método de junção de grupos, o produto interno dos vetores das velocidades das melhores partículas dos subenxames em questão deve ser calculado. Se este for menor do que zero, ambos estão movendo-se na mesma direção. Sendo assim, o subenxame como o menor G_{best} terá suas partículas reposicionadas aleatoriamente no espaço de busca, sendo reincluídas no enxame principal. Nota-se que este método não mais pode ser chamado de método de fusão, mas de espalhamento.

No novo método de absorção de partículas em subenxames já existentes, os subenxames passarão a permitir ou não novas inclusões. A diversidade de um suben-

xame deve ser calculada através da distância média entre o G_{best} do grupo e as demais partículas da população. Se este valor for menor do que um limiar definido previamente pelo usuário, novas partículas serão aceitas nesse subenxame.

O NichePSO é descrito, resumidamente, no Algoritmo 2, adaptado do trabalho de Madeiro [3].

Algoritmo 2: Algoritmo do NichePSO.

```
Inicia as partículas do enxame principal;
enquanto Critério de parada não for alcançado faça
    Treina as partículas no enxame principal;
    Atualiza o fitness de cada partícula no enxame principal;
    para cada Subenxame formado faça
        Treina as partículas utilizando as Equações (2.15) e 2.14;
        Atualiza a aptidão das partículas;
        Atualiza o raio do subenxame;
    Executa a junção de subenxames se necessário;
    Remove partículas do enxame principal que adentraram na região correspondente a
    um subenxame já existente e as adiciona a tal subenxame;
    Procura no enxame principal por partículas que satisfaçam a condição de criação de
    uma nova subpopulação e cria as subpopulações correspondentes;
Retorna as melhores partículas de cada subenxame como soluções do problema.
```

Na Equação (2.15), $v_{ij}(t+1)$ é a nova velocidade da partícula i na dimensão j, $v_{ij}(t)$ é o valor da mesma dimensão da velocidade antiga da mesma partícula, $c1$ e $c2$ são constantes definidas pelo usuário, $r1$ e $r2$ são números aleatórios entre 0 e 1, $G_{best_j}(t)$ e $P_{best_{ij}}(t)$ são os melhores valores para a dimensão j encontrados pela população e pela partícula j até o momento, respectivamente, e $x_{ij}(t)$ é o valor do componente j do vetor de posição da partícula i no tempo t.

$$v_{ij}(t+1) = v_{ij}(t) + c1r1(G_{best_j}(t) - x_{ij}(t)) + c2r2(P_{best_{ij}}(t) - x_{ij}(t)). \tag{2.15}$$

Glowworm Swarm Optimization

Proposto por Krishnanand e Ghose [11], o algoritmo *Glowworm Swarm Optimization* (GSO) baseia-se no comportamento de um enxame de vaga-lumes no momento da formação de casais para acasalamento. Na natureza, os machos utilizam da emissão de luz para atrair as fêmeas, sendo os detentores de maior intensidade os mais atraentes.

No algoritmo, os indivíduos (vagalumes) possuem como atributo o nível de luciferina, o qual é calculado de acordo com a Equação (2.16), onde $l_i(t+1)$ e $l_i(t)$ são os níveis de luciferina atualizado e antigo, respecivamente, ρ é a taxa de decaimento deste nível ($0 \leq \rho \leq 1$), $J(x_i(t))$ é o valor da função objetivo do vaga-lume i no tempo t e γ é um ponderador do valor da função que entrará neste cálculo.

$$l_i(t+1) = (1-\rho)l_i(t) + \gamma J(x_i(t)). \tag{2.16}$$

O campo de percepção de um vaga-lume i é delimitado por um raio de comprimento r_i, sendo, dessa forma, a vizinhança de i formada pelos indivíduos cujas distâncias para este são menores do que o r_i e suas aptidões sejam maiores do que $J(x_i)$. Com o objetivo de controlar o número de vizinhos dos indivíduos, o raio de percepção é ajustado durante a execução do algoritmo. Se o número de vizinhos do indivíduo i for maior do que um limiar n_t, o seu raio de percepção r_i diminuirá. Porém, caso o número de vizinhos seja menor do que este limiar, o raio aumentará até um limite r_s. Formalmente, o processo de ajuste do raio é descrito pelas Equações (2.17) e (2.18), onde $r_i(t)$ e $r_i(t+1)$ são os raios antes e depois da atualização, $|N_i(t)|$ é o número de vaga-lumes vizinhos no tempo t e d_{ij} é a distância Euclidiana entre os vaga-lumes i e j.

$$r_i(t+1) = min\{r_s, max\{0, r_i(t) + \beta(n_t - |N_i(t)|)\}\}, \tag{2.17}$$

$$N_i(t) = \{j : d_{ij} < r_i(t); l_i(t) < l_j(t)\}. \tag{2.18}$$

Após a atualização do nível de luciferina e antes da atualização do raio de percepção, os vaga-lumes escolherão qual dos seus vizinhos deverão seguir na fase de movimentação. A escolha do vaga-lume i é feita em cima da função de probabilidade descrita pela Equação (2.19), onde $p_{ij}(t)$ é a probabilidade do vaga-lume i escolher o indivíduo j como guia no tempo t, $l_j(t)$, $l_i(t)$ e $l_k(t)$ são os níveis de luciferina dos vaga-lumes i, j e k, respectivamente, no mesmo momento, $N_i(t)$ é conjunto de vaga-lumes vizinhos do indivíduo i e $j \in N_i(t)$. O indivíduo deve, logo após a seleção do seu guia, mover-se na direção do mesmo, como mostra a Equação (2.20), onde $x_{id}(t)$ e $x_{id}(t+1)$ são os valores do componente d do vetor de posição do vaga-lume i antes e depois da movimentação, respectivamente, $x_{jd}(t)$ é o valor do mesmo componente no indivíduo j no momento da execução da movimentação, $norma_euclidiana()$ representa

uma função que retorna norma euclidiana e s é o tamanho do passo que o vaga-lume i dará na direção do indivíduo j.

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)}, \tag{2.19}$$

$$x_{id}(t+1) = x_{id}(t) + s\left(\frac{x_{jd}(t) - x_{id}(t)}{norma_euclidiana(x_{jd}(t) - x_{id}(t))}\right). \tag{2.20}$$

Se um indivíduo não possui nenhum outro vaga-lume como vizinho, este vai aguardar um outro vaga-lume mais apto entrar no seu raio de percepção para mover-se. Enquanto isto não ocorre, este permanecerá estacionado. Ao executar o algoritmo, observa-se uma alternância de "líderes" de grupos de vaga-lumes, sendo tal fenômeno chamado de "efeito *leapfrogging*".

O algoritmo GSO está resumido no Algoritmo 3.

Algoritmo 3: Algoritmo do GSO.

```
Inicializa os vaga-lumes na população;
enquanto As condições de parada não são satisfeitas faça
    para cada Vaga-lume faça
        Atualiza o nível de luciferina usando (2.16);
    para cada Vaga-Lume faça
        Atualiza o conjunto de vizinhos;
        Calcula a probabilidade de escolha do guia entre seus vizinhos usando 2.19;
        Seleciona vaga-lume guia;
        Executa movimentação usando (2.20);
        Atualiza o raio de percepção usando (2.17) e (2.18);
Retorna melhor solução.;
```

Density based Fish School Search

Nesta seção, será apresentada uma já existente abordagem para otimização de multi-solução baseada no algoritmo *Fish School Search*, o *Density based Fish School Search (dFSS)* o qual foi proposto por Madeiro, Bastos-Filho e Lima Neto [3][12]. Neste algoritmo, os peixes possuem a capacidade de armazenar determinadas informações, isto com o objetivo de segregar cardumes em subcardumes, fazendo com que estes explorem diferentes nichos à procura das soluções do problema. Dessa forma, cada subcardume torna-se reponsável por explorar e manter estas regiões, as

Algoritmo 4: Algoritmo do dFSS.

```
Os peixes são inicializados em posições aleatórias com seus pesos iguais a zero;
Avalia as aptidões dos peixes;
Calcula as distâncias entre os peixes;
enquanto Critério de parada não for alcançado faça
    para cada Peixe do cardume faça
        Executa o Movimeto Individual descrito em 2.2.3;
        se A aptidão do peixe aumentou então
            para cada Peixe no cardume faça
                Executa o Operador de Alimentação descrito em 2.2.3;
                Atualiza o peso do peixe;
                Executa o Operador de Memória descrito em 2.2.3;
    para cada Peixe i do cardume faça
        Executa o Movimento Coletivo Instintivo para i descrito em 2.2.3; Determina o
        peixe de maior influência para i;
    Executa o Operador de Divisão do Cardume Principal descrito em 2.2.3;
    para cada Subcardume i gerado faça
        Calcula o baricentro de i;
    para cada Peixe do cardume faça
        Atualiza o tamanho do passo individual;
        Executa o Movimento Coletivo Volitivo do peixe em questão descrito em 2.2.3;
    Avalia a aptidão de cada peixe do cardume;
    Calcula as distâncias entre os peixes;
    Atualza valor de decay_max(t), operação descrita em 2.2.3;
Retorna os melhores indivíduos de cada subcardume formado na última iteração como
soluções localizadas pelo algoritmo.
```

quais possuem potenciais soluções para o problema.

Foram acrescentados dois operadores ao FSS para a criação do dFSS: o Operador de Memória ou Afinidade e o Operador de Divisão do Cardume. Porém, os operadores já existentes também foram modificados para que os objetivos da busca fossem alcançados. O algoritmo dFSS é descrito pelo Algoritmo 4, onde é possível verificar a ordem de execução dos operadores. A seguir, tais operadores serão detalhados para um melhor entendimento dessa abordagem.

Movimento Individual O mecanismo de Movimento Individual manteve-se igual em relação ao FSS original. Porém, o processo de atualização do tamanho do passo deste operador foi alterado com o objetivo de formar subcardumes para explorar diferentes nichos, além de mantê-los durante a execução do algoritmo. Este mecanismo

de atualização é descrito pelas Equações (2.21), (2.22), (2.23) e (2.24), onde $step_{ind_i}(t)$ e $step_{ind_i}(t+1)$ são os tamanhos do passo individual antes e depois da atualização, $decay_i(t)$ é a taxa de decaimento no tempo t, $decay_{min}$ e $decay_{max}$ são as taxas de decaimento mínimo e máximo, respectivamente, $decay_{max_{ini}}$ e $decay_{max_{fim}}$ são as taxas de decaimento máximo inicial e final, respectivamente, T_{max} é o número de iterações do algoritmo, Δf_i é a variação do *fitness* do peixe i na última iteração, q_{ij} é a quantidade de indivíduos entre o peixe i e o peixe j (indivíduos cujas distâncias para o peixe i são menores do que a distância de i para j), incluindo o peixe i, e $d_{R_{jk}}$ representa a distância relativa entre os peixes j e k, ou seja, $d_{R_{jk}} = \frac{d_{jk}}{mind_{ij}}$, onde j representa todos os outros peixes diferentes de i. A taxa de decaimento deve estar entre 0 e 1, sendo este valor adaptado de acordo com a quantidade de alimento encontrado. Dessa forma, os peixes que encontrarem uma maior quantidade de alimento devem ter seus comprimentos do passo individual mais encurtados do que os outros indivíduos. Assim, a localização dos peixes com maior sucesso no processo de busca é conservada, mantendo uma exploração nos nichos já encontrados.

$$step_{ind_i}(t+1) = decay_i(t) * step_{ind_i}(t), \tag{2.21}$$

$$decay_i(t) = decay_{min} - \left(\frac{R_i(t) - min(R_j(t))}{max(R_j(t)) - min(R_j(t))}\right)(decay_{min} - decay_{max}(t)), \tag{2.22}$$

$$decay_{max}(t) = decay_{max_{ini}}\left(\frac{decay_{max_{fim}}}{decay_{max_{ini}}}\right)^{t/T_{max}}, \tag{2.23}$$

$$R_i(t) = \sum_{j=1}^{Q} \frac{\Delta f_i}{(d_{R_{ij}})^{q_{ij}} \sum_{k=1}^{N} \frac{1}{(d_{R_{jk}})^{q_{jk}}}}. \tag{2.24}$$

Operador de Alimentação Assim como no FSS original, no dFSS a variação de fitness obtida no movimento individual representa a quantidade de alimento ingerido pelo peixe em questão. Porém, nessa abordagem, será introduzido o conceito da partilha de alimentos. Este operador é descrito pela Equação (2.25), onde $C(i,j)$ é a quantidade de alimento partilhado de i para j, Δf_i é a variação do *fitness* do peixe i na última iteração, q_{ij} é a quantidade de indivíduos entre o peixe i e o peixe j (indivíduos cujas distâncias para o peixe i são menores do que a distância de i para j), incluindo o

peixe i, e $d_{R_{jk}}$ representa a distância relativa entre os peixes j e k, ou seja, $d_{R_{jk}} = \frac{d_{jk}}{mind_{ij}}$, onde j representa todos os outros peixes diferentes de i. Tal partilha possui como objetivo o equilíbrio entre a competição e a cooperação entre os peixes na busca por fontes de alimentos. Dessa forma, a quantidade de alimento encontrado por um peixe i partilhada com outros membros do cardume depende da distância relativa do peixe i ao peixe receptor da parcela do alimento e da densidade de peixes em torno de i (quantidade de peixes em torno deste). O alimento ingerido por cada peixe ao final desta operação corresponde à quantidade de alimento encontrado pelo mesmo e todas os particionamentos de alimentos feitos pelos peixes bem sucedidos e seus movimentos individuais.

$$C(i,j) = \frac{\Delta f_i}{(d_{R_{ij}})^{q_{ij}} \sum_{k=1}^{N} \frac{1}{(d_{R_{jk}})^{q_{jk}}}} \tag{2.25}$$

Operador de Memória O Operador de Memória tem como objetivo permitir que os peixes memorizem quais indivíduos compartilharam uma maior quantidade de alimento durante o processo de busca, sendo os maiores compartilhadores os mais seguidos. Ou seja, para o peixe i, quanto maior for a quantidade de alimento compartilhado M_{ij} pelo peixe j, maior será influência do peixe j no comportamento do peixe i. A memória de um indivíduo é representada por um vetor $M_i = \{M_{i1}, M_{i2}, ..., M_{iN}\}$, onde N é a quantidade de peixes pertencentes ao cardume. Durante a partilha dos alimentos realizada por cada peixe, as quantidades de alimento compartilhado são somados aos elemento correspondentes nos vetores de cada indivíduo. Este processo é descrito pela Equação (2.26), onde $M_{ij}(t)$ e $M_{ij}(t+1)$ são os valores da contribuição do peixe j com a alimentação do peixe i antes e depois da atualização, respectivamente, $0 \leq \rho \leq 1$, sendo ρ uma taxa de esquecimento, e $C(j,i)$ a contribuição de j para i descrita pela Equação (2.25).

$$M_{ij}(t+1) = (1-\rho)M_{ij}(t) + C(j,i). \tag{2.26}$$

Movimento Coletivo Instintivo No Movimento Coletivo Instintivo do FSS original, todos os peixes locomovem-se na mesma direção. No dFSS, cada peixe possui sua direção de movimento, a qual é definida pelos indivíduos mais influentes do peixe em questão, sendo os peixes mais bem-sucedidos os mais importantes na definição

dessas direções. Como pode ser observado na Equação (2.27), onde $x_{ij}(t)$ e $x_{ij}(t+1)$ são os valores dos componentes j dos vetores das posições dos peixes i antes e depois da atualização, respectivamente, N é o número de indivíduos no cardume, Δx_{kj} é a variação do valor do componente j do vetor da posição do peixe k após o Movimento Individual e $M_{ik}(t)$ é a o valor da contribuição do peixe k na alimentação do peixe i durante o processo de busca, a movimentação é feita através da soma de uma média ponderada aos componentes do vetor de posição de um peixe i, onde os pesos desta média são as contribuições dadas a tal indivíduo por cada peixe, onde quanto maior for a contribuição de um peixe j, maior influência este irá exercer sobre o peixe i.

$$x_{ij}(t+1) = x_{ij}(t) + \left(\frac{\sum_{k=1}^{N} \Delta x_{kj} M_{ik}(t)}{\sum_{k=1}^{N} M_{ik}(t)} \right). \tag{2.27}$$

Operador de Divisão do Cardume Principal Este operador é responsável pela divisão do cardume principal em subcardumes. Um peixe i faz parte do mesmo subcardume de um peixe j se e somente se i for o peixe mais influente para j ou j for o indivíduo mais influente para i. Apesar destas condições não serem simétricas, ou seja, i pode ser o peixe mais influente para j mas o contrário pode não ser verdadeiro, se i faz parte do mesmo subcardume j, j faz parte do mesmo subcardume de i. O processo de definição de subcardumes começa com a escolha aleatória e remoção de um peixe i do cardume. Todos os peixes j pertencentes ao mesmo subcardume do peixe i são também removidos. Em seguida, todos os peixes k pertencentes ao mesmo subcardume de todos os peixes j são também removidos do cardume. Este processo ocorre até que todos os peixes pertencentes ao subcardume de i tenham sido removidos do cardume principal. Após essa condição ser satisfeita, um novo peixe é escolhido aleatoriamente e removido do cardume principal, repetindo a mesma operação, até que o cardume principal fique vazio. Ao final da execução deste procedimento, deve haver uma lista com todos os subcardumes formados.

Movimento Coletivo Volitivo O Movimento Coletivo Volitivo possui como objetivo facilitar a convergência dos subcardumes para os nichos encontrados, mantendo-os no processo de busca. Diferente do FSS original, no dFSS este movimento ocorre internamente a cada subcardume. A movimentação em direção ao baricentro ocorre segundo a Equação (2.28), onde $x_{ij}(t)$ e $x_{ij}(t+1)$ são os valores do componente j dos

vetores de posição do indivíduo i antes e depois da atualização, respectivamente, $0 \leq decay_{max}(t) \leq 1$ e é definido pela Equação (2.23) e B_{kj} é o componente j do baricentro do subcardume k.

$$x_{ij}(t+1) = x_{ij}(t) + (1 - decay_{max}(t))(B_{kj}(t) - x_{ij}(t)). \tag{2.28}$$

Na próxima seção, uma nova heurística para segregação de cardumes no algoritmo FSS será apresentada. Esta abordagem visa a criação de uma base para a proposta futura de um novo método de otimização multi-solução.

3 *Utilizando o Peso do Peixe como Elemento Segregador*

Neste capítulo, estão apresentadas novas heurísticas para segregação populacional aplicadas ao algoritmo *Fish School Search*. Esta proposta visa embasar o posterior desenvolvimento de um novo algoritmo de otimização multi-solução de funções multimodais.

Nesta abordagem, o algoritmo faz uso do peso dos peixes como operador de segregação populacional. Com a definição de subcardumes através da criação de relações "guia-guiado" entre os indivíduos, um nível de independência diretamente proporcional ao peso do indivíduo emergirá do sistema, onde os peixes mais pesados executarão movimentos de forma mais independente do que os peixes mais leves, sendo estes últimos guiados pelos mais pesados, ou seja, pelos indivíduos que obtiveram um maior sucesso em seu histórico de busca.

Nas seções a seguir estão descritos os operadores utilizados nesta abordagem. Na Seção 3.1, uma visão geral do algoritmo é apresentada. Nas Seções 3.2, 3.3 e 3.4 são explanados todos os operadores após a devidas modifcações.

3.1 Visão Geral do Algoritmo Proposto

Neste trabalho, propõe-se o acréscimo de um novo operador ao algoritmo FSS (versão inicial, Vanilla), além de modificações necessárias feitas nos operadores já existentes. O novo operador é chamado de Operador de Definição de Subcardumes. Uma versão simples do algoritmo está apresentada no Algoritmo 5.

Após a inicialização de todos os parâmetros do algoritmo, o *loop* principal é executado. O primeiro operador a ser executado em cada iteração é o Operador de Definição de Subcardumes. Neste operador, as relações "guia-guiada" são estabelecidas

Algoritmo 5: Algoritmo simplificado do FSS modificado.

```
1 Parâmetros são inicializados;
2 enquanto Condição de parada não é alcançada faça
3     Define subcardumes para a iteração corrente;
4     Executa Movimento Individual;
5     Executa Operador de Alimentação;
6     Executa Operador de Movimento Coletivo Instintivo;
7     Executa Operador de Movimento Coletivo Volitivo;
8 Retorna melhor solução;
```

baseando-se nos pesos dos indivíduos. Os indivíduos mais pesados possuem uma maior probabilidade de serem guias, enquanto os mais leves devem ser os guiados. Após este passo, os próximos a serem executados já existem do FSS original: Movimento Individual, Alimentação, Movimento Coletivo Instintivo e Movimento Coletivo Volitivo. O movimento individual e o processo de alimentação não sofreram alterações em relação ao FSS original. Alterações foram feitas nos operadores de movimentos coletivos instintivo e volitivo com o objetivo de permitir que os subcardumes convirjam e explorem os seus nichos encontrados durante o processo de busca.

Nas próximas seções, tais operadores serão detalhados para um entendimento completo desta abordagem.

3.2 Operador de Definição de Subcardumes

O Operador de Definição de Subcardumes é responsável por definir relações hierárquicas entre os peixes, através das quais será realizada a comunicação entre indivíduos. Pode-se denominar tal relação binária como "*guia-guiado*", onde o guia passa a exercer uma maior influência na movimentação do guiado. Dois indivíduos que possuem esta relação são chamados de *companheiros*. O peixe mais pesado é considerado guia do peixe mais leve, sendo este o guiado, sendo tais papéis assumidos apenas nos movimentos coletivos. Este operador é descrito no Algoritmo 6. Como pode ser observado, cada peixe, em ordem aleatória, deve ser confrontado com todos os outros indivíduos e, para cada um, deve-se decidir se será ou não escolhido como guiado. Por conta das variáveis C_r e C_i na Equação (3.1), as quais podem variar a cada nova ligação feita, a ordem dos peixes a serem iterados e dos indivíduos a escolherem seus guiados influencia nos resultados.

Algoritmo 6: Operador de Definição de Subcardumes.

```
1  S:Conjunto de peixes do cardume principal;
2  T:Conjunto temporário de peixes do cardume principal;
3  Remove todas as antigas ligações entre todos os peixes;
4  enquanto S não é vazio faça
5      Sorteia um peixe i em S e o remove do mesmo;
6      Reinicia T com todos os peixes da população;
7      enquanto T não é vazio faça
8          Sorteia um peixe r em T e o remove do mesmo;
9          se i é diferente de r então
10             se p_ir(t) calculado pela Equação (3.1) é maior ou igual a um valor aleatório
               gerado entre 0 e 1 e r já não foi escolhido como companheiro de i nem o
               contrário então
11                 i é companheiro de r e vice-versa, sendo i guia de r e r guiado de i;
```

$$p_{ir} = \frac{W_i(t)}{W_r(t)C_rC_i}. \tag{3.1}$$

Na Equação (3.1), p_{ir} é a razão de dominância do peixe i sobre o peixe r, onde $W_r(t)$ e $W_i(t)$ são os pesos dos peixes r e i no tempo t, C_r é o número de companheiros do peixe r no momento exato da consulta e C_i é o mesmo atributo do peixe i no mesmo instante.

Nota-se que a probabilidade de escolha de um peixe r como guiado de i é inversamente proporcional ao peso de r e diretamente proporcional ao peso de i. Quanto maior o peso de um peixe mais improvável será a escolha do mesmo para ser guiado por outro indivíduo. Por outro lado, quanto maior for este valor, maior será a probabilidade deste peixe ser guia de outro(s). Porém, para evitar o monopólio de determinados peixes pesados no papel de guia, a probabilidade de um peixe escolher um novo guiado é inversamente proporcional ao número de companheiros já escolhidos durante a execução deste operador. Em outras palavras, à medida que o número de peixes guiados por um determinado indivíduo cresce, a probabilidade deste escolher um novo guiado decresce. Para evitar que os mesmos peixes sejam escolhidos para serem guiados por serem muito leves, a probabilidade de um peixe ser escolhido para ser guiado é também inversamente proporcional ao número de companheiros do mesmo. Dessa forma, à medida que o número de companheiros aumenta, a probabilidade deste peixe ser escolhido como guiado diminui. Em resumo, quanto mais ligações dois peixes possuem, menor será a probabilidade destes serem interligados

entre si. Tais operações são executadas para permitir que uma maior exploração do espaço de busca feita por pequenos grupos, objetivando encontrar o maior número de soluções possível.

Considerando uma indexação fixa no vetor dos peixes da população e uma varredura seqüencial neste vetor, os primeiros peixes, ou seja, os peixes com os menores índices, possuem uma maior probabilidade de escolha, pois no início do processo de seleção de companheiros, após todas as ligações entre os indivíduos serem removidas, os valores de C_r e C_i estão ainda baixos. Dessa forma, os primeiros peixes do vetor possuiriam uma maior probabilidade de serem escolhidos. Além disso, os primeiros peixes a realizarem suas escolhas possuiriam uma maior chance de obtenção de novos companheiros, pois muitos dos indivíduos ainda não haveriam sido escolhidos. Por isso, como pode ser observado no algoritmo, a ordem dos peixes a escolherem seus guiados deve ser aleatória, bem como a ordem dos peixes a serem iterados durante o processo de escolha. Desta forma, qualquer dependência da ordem dos peixes no vetor representante da população no processo de definição de companheiros é anulada.

3.3 Operador de Movimento Coletivo Instintivo

Neste operador, o algoritmo herdou as bases do operador de mesmo nome do FSS original. Porém, na abordagem proposta neste trabalho, cada peixe deve executar seu Movimento Coletivo Instintivo levando em consideração apenas os seus guias e a si mesmo. A Equação (3.2) define o novo Operador de Movimento Coletivo Instintivo, onde $x_{ij}(t)$ é o valor da dimensão j do vetor de posição do peixe i no tempo t, $x_{ij}(t+1)$ é tal valor atualizado, ou seja, no tempo $t+1$, N_g é o número de peixes guia do peixe i na iteração corrente, Δx_{kj} é a variação do valor da dimensão j do vetor de posição do peixe k pertencente ao conjunto de peixes guia do peixe i antes e depois do último Movimento Individual e $\Delta f(x_k)$ é variação do fitness do peixe k pertencente ao conjunto de peixes guia do peixe i.

$$x_{ij}(t+1) = x_{ij}(t) + \frac{\Delta x_{ij}\Delta f(x_i) + \sum_{k=1}^{N_g} \Delta x_{kj}\Delta f(x_k)}{\Delta f(x_i) + \sum_{k=1}^{N_g} \Delta f(x_k)}. \tag{3.2}$$

Nota-se através da Equação (3.2) que cada peixe possui seu vetor de movimentação no movimento coletivo instintivo. Sendo o cálculo deste vetor limitado aos guias de

cada indivíduo, através deste operador os peixes mais pesados guiam os mais leves para regiões promissoras do espaço de busca. Além disso, mesmo que a relação do tipo guia-guiado seja simétrica, devido ao fato deste vetor de movimentação ser calculado levando em consideração apenas os peixes guias, tais peixes não serão afetados pelos peixes guiados, pois estes não são considerados neste cálculo.

3.4 Operador de Movimento Coletivo Volitivo

Assim como ocorre no Operador de Movimento Coletivo Instintivo, o volitivo herdou os mecanismos básicos do operador homônimo do FSS original, sendo a única diferença o fato de que cada indivíduo deve calcular um "baricentro pessoal", sendo este definido através dos pesos e das posições dos companheiros e do próprio peixe em questão, ou seja, dos indivíduos com os quais este possui uma relação guia-guiado, independente de qual dos dois papéis este tenha assumido. O cálculo do baricentro é explanado na Equação (3.3), onde $B_{ij}(t)$ é o valor da componente j do baricentro pessoal do peixe i no tempo t, $x_{ij}(t)$ é o valor do componente j do vetor de posição do indivíduo i no tempo t, $W_i(t)$ é o peso do indivíduo i no tempo t, $x_{kj}(t)$ é o valor do componente j do vetor de posição do indivíduo k no tempo t, $W_k(t)$ é o peso do indivíduo k no tempo t e N^i_{comp} é o número de companheiros do peixe em questão.

$$B_{ij}(t) = \frac{x_{ij}(t)W_i(t) + \sum_{k=1}^{N^i_{comp}} x_{kj}(t)W_k(t)}{W_i(t) + \sum_{k=1}^{N^i_{comp}} W_k(t)}. \tag{3.3}$$

No novo algoritmo, os mecanismos de contração e expansão, de certa forma, ocorrem internamente aos subcardumes. Apesar de cada peixe ter seu baricentro calculado a através dos seus companheiros, sendo a relação guia-guiado simétrica, o guiado exerce influência sobre o cálculo do baricentro pessoal do guia e vice-versa. Dessa forma, existe uma atração ou repulsão mútua entre tais indivíduos no momento da contração e expansão, respectivamente, mesmo que esses efeitos sejam atenuados por outros indivíduos que façam parte da mesma rede de relações guia-guiado. Os mecanismos de contração (Equação (3.4)) e expansão (Equação (3.5)) ocorrem se o peso aumentar ou não aumentar, respectivamente. A função $distance(\vec{A}, \vec{B})$ retorna a distância euclidiana entre dois vetores $\vec{A}$ e $\vec{B}$.

$$x_{ij}(t+1) = x_{ij}(t) - step_{vol} rand(0,1) \frac{(x_{ij}(t) - B_{ij}(t))}{distance(x_i(t), B_{ij}(t))} \tag{3.4}$$

$$x_{ij}(t+1) = x_{ij}(t) + step_{vol} rand(0,1) \frac{(x_{ij}(t) - B_{ij}(t))}{distance(x_i(t), B_{ij}(t))} \tag{3.5}$$

4 Descrição dos Experimentos

Neste capítulo, a metodologia utlizada na realização dos experimentos é descrita. Foram utilizados três algoritmos de otimização multi-solução para comparar com a abordagem aqui proposta: NichePSO, GSO e dFSS. Na Seção 4.1, o ajuste dos parâmetros de configuração de cada técnica é detalhado. Na Seção 4.2, é explanada a definição de solução utilizada na nova abordagem. Na Seção 4.3, são expostas todas as sete funções de *benchmark* utilizadas. Por fim, na Seção 4.4, a configuração dos experimentos é detalhada.

4.1 Configuração dos Parâmetros

As configurações do NichePSO, GSO e dFSS foram as mesmas utilizadas por Madeiro [3] em sua proposta do dFSS. Os resultados evocados serão comparados com a proposta deste trabalho. Nas Tabelas 4.1, 4.2, 4.3 e 4.4, estão apresentadas as configurações paramétricas dos algoritmos NichePSO, GSO, dFSS e também da nova abordagem proposta neste trabalho, respectivamente, as quais foram utilizadas em todos os experimentos deste trabalho. Os parâmetros utilizados na nova abordagem foram definidos experimentalmente, sendo estes os parâmetros responsáveis por retornar os melhores resultados.

Tabela 4.1: Configuração Paramétrica do NichePSO.

$c1$	$c2$	ω	δ	μ	ε
1,2	1,2	Linearmente decrescente de 0,7 a 0,2	10^{-4}	10^{-2}	0,1

Tabela 4.2: Configuração Paramétrica do GSO.

ρ	γ	β	n_t	s	l_0
0,4	0,6	0,08	5	0,03	5

Tabela 4.3: Configuração Paramétrica do dFSS.

ρ	$step_{init}$	$decay_{min}$	$decay_{max_{init}}$	$decay_{max_{end}}$
0,3	0,05	0,999	0,99	0,95

Tabela 4.4: Configuração Paramétrica da FSSm.

$step_{ind}$	$step_{vol}$
Linearmente decrescente de 0,4 a 0	Linearmente decrescente de 0,025 a 0

4.2 Definição de Solução

Para esta abordagem, uma solução retornada pelo algoritmo é definida pelo melhor indivíduo de cada subcardume no final da execução do algoritmo. No entanto, ao final da execução da última iteração, os subenxames são redefinidos utilizando um método diferente do qual foi utilizado durante todo o processo de busca. Este método consiste em considerar do mesmo subenxame dois peixes cujas distâncias entre si seja menor ou igual a 0,01. Tal valor foi atribuido experimentalmente, sendo este considerado um valor razoável para que o algoritmo possa reduzir o número total de soluções retornadas, mas sem considerar dois peixes de nichos diferentes como pertencentes do mesmo subcardume.

Retornadas as soluções, deve ser verificado quais destas correspondem a alguma solução da função. Considera-se que uma solução $\overrightarrow{k}$ da função foi encontrada corretamente se a distância normalizada (Equações (4.1) e (4.2), onde D é o número de dimensões) da solução retornada pelo algoritmo e $\overrightarrow{k}$ for menor do que 0,005, sendo tal valor utilizado por Madeiro [3] nos experimentos com o dFSS. Madeiro também definiu que uma solução retornada pelo algoritmo é considerada incorreta se a menor distância para qualquer solução verdadeira da função for maior do que 0,01. Tais valores foram repetidos neste trabalho para que as comparações sejam feitas de forma justa.

$$d_N(\overrightarrow{i}, \overrightarrow{j}) = \sqrt{\frac{(x_N^i - x_N^j) \cdot (x_N^i - x_N^j)}{D}}, \tag{4.1}$$

$$x_N = \left(\frac{x_1}{x_{1_{max}}}, \frac{x_2}{x_{2_{max}}}, ..., \frac{x_D}{x_{D_{max}}} \right). \tag{4.2}$$

4.3 Funções de Teste

Nesta seção, são apresentadas as sete funções de teste utilizadas nos experimentos deste trabalho.

4.3.1 Equal Peaks A

Esta função está representada pela Equação (4.3) (m é o número de dimensões) e ilustrada pela Figura 4.1 em sua versão de 2 dimensões. Ela possui diversos picos idênticos distribuidos uniformemente no espaço de busca. Neste trabalho, foi utilizada uma versão desta função onde $m = 2$.

$$f(\overrightarrow{X}) = \sum_{i=1}^{m} cos^2(X_i). \tag{4.3}$$

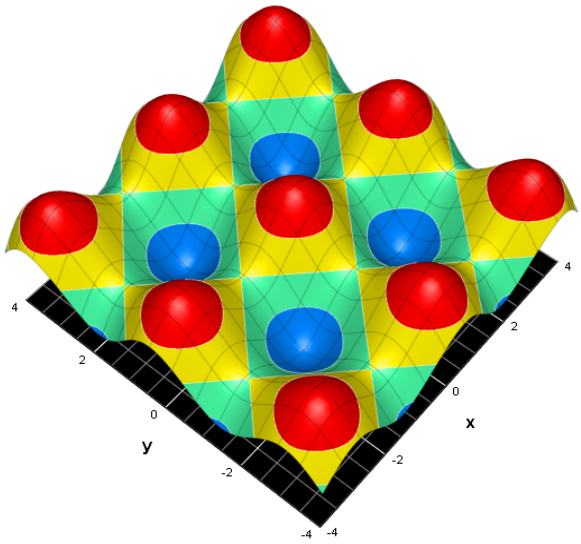


Figura 4.1: Superfície da versão bidimensional da função Equal Peaks A.

4.3.2 Equal Peaks B

Equal Peaks B é representada pela Equação (4.4) e ilustrada pela Figura 4.2. Assim como Equal Peaks A, possui picos idênticos distribuidos uniformemente no espaço de busca. Porém, estes estão mais próximos uns dos outros.

$$f(x,y) = cos^2(x) + sin^2(y). \tag{4.4}$$

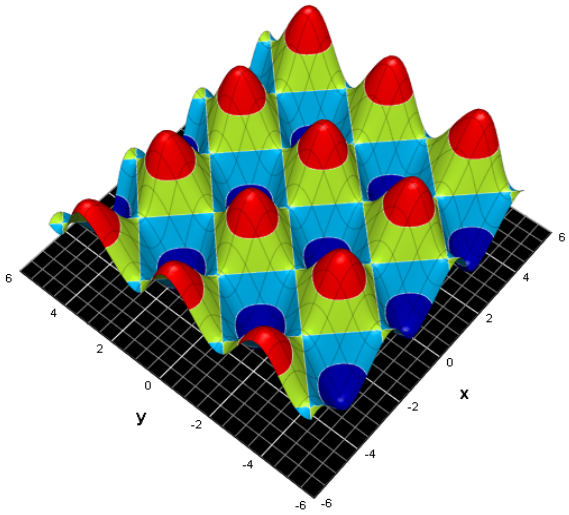


Figura 4.2: Superfície da função Equal Peaks B.

4.3.3 Griewank

Griewank é representada pela Equação (4.5) e ilustrada pela Figura 4.3. Esta função apresenta uma quantidade muito maior de picos distribuidos no espaço de busca, o que dificulta o isolamento dos subenxames entre si. Além disso, os picos mais distantes dos centros são mais altos do que os que estão próximos à origem, o que dificulta a uma distribuição mais uniforme dos indivíduos ao longo das demais soluções, concentrando-os nestes picos mais altos. Neste trabalho, foi utilizada uma versão desta função onde $m = 2$, onde m é o número de dimensões.

$$f(\overrightarrow{X}) = 1 + \sum_{i=1}^{m} \frac{x_i^2}{4000} - \prod_{i=1}^{m} cos\left(\frac{x_i}{\sqrt{i}}\right). \tag{4.5}$$

4.3.4 Himmelblau

Himmelblau possui apenas quatro picos suaves e é representada pela Equação (4.6) e ilustrada pela Figura 4.4.

$$f(x,y) = 200 - (x^2 + y^2 - 11)^2 - (x + y^2 - 7)^2. \tag{4.6}$$

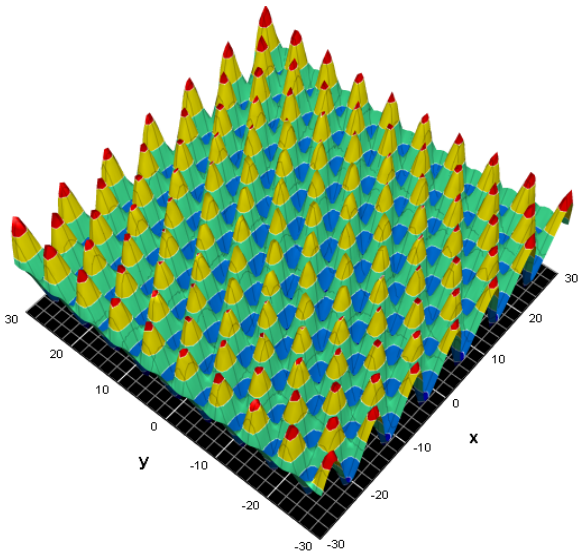


Figura 4.3: Superfície da versão bidimensional da função Griewank.

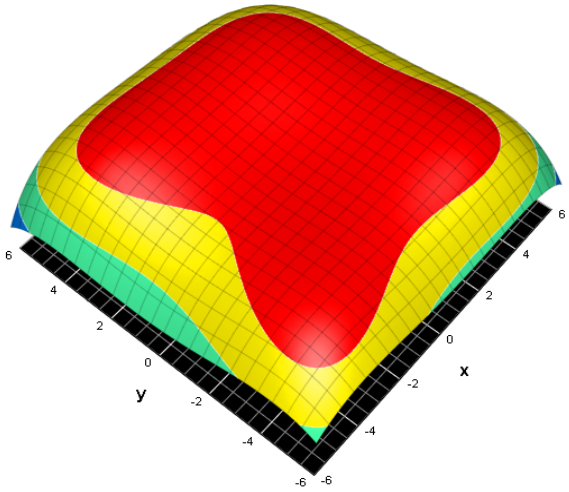


Figura 4.4: Superfície da função Himmelblau.

4.3.5 Peaks

Peaks possui três máximos locais. Esta função é representada pela Equação (4.7) e ilustrada pela Figura 4.5.

$$f(x,y) = 3(1-x)^2 e^{-[x^2+(y+1)^2]} - 10\left(\frac{x}{5} - x^3 - y^5\right) e^{-(x^2+y^2)} - \frac{1}{3} e^{-[(x+1)^2+y^2]}. \quad (4.7)$$

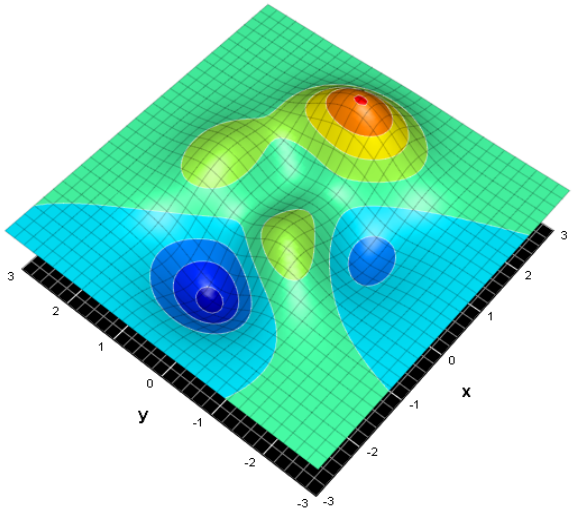


Figura 4.5: Superfície da função Peaks.

4.3.6 Random Peaks

Random Peaks, representada pela Equação (4.8), possui Q soluções representados por (x_i, y_i), sendo estes distribuídos aleatoriamente no espaço de busca. Na Figura 4.6, a função possui dez soluções. Na Equação (4.8), $a_i = 1 + 2\vartheta$, $b_i = 2 + \vartheta$, $x_i = -5 + 10\vartheta$, $y_i = -5 + 10\vartheta$ e ϑ é um valor gerado aleatoriamente entre zero e um.

$$f(x,y) = \sum_{i=1}^{Q} a_i e^{-b_i[(x-x_i)^2+(y-y_i)^2]}. \tag{4.8}$$

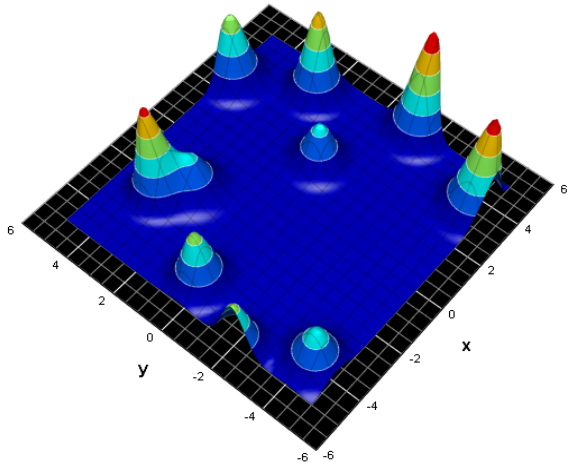


Figura 4.6: Superfície da função Random Peaks.

4.3.7 Rastrigin

Rastrigin é representada pela Equação (4.9) e ilustrada pela Figura 4.7. Assim como Griewank, esta função apresenta uma quantidade muito maior de picos distribuidos no espaço de busca, o que dificulta o isolamento dos subenxames entre si. Possui a mesma característica dos picos mais altos se concentrarem nos extremos. Neste trabalho, foi utilizada uma versão desta função onde $m = 2$, onde m é o número de dimensões.

$$f(\overrightarrow{X}) = 10m + \sum_{i=1}^{m}[x_i^2 - 10cos(2\pi x_i)]. \tag{4.9}$$

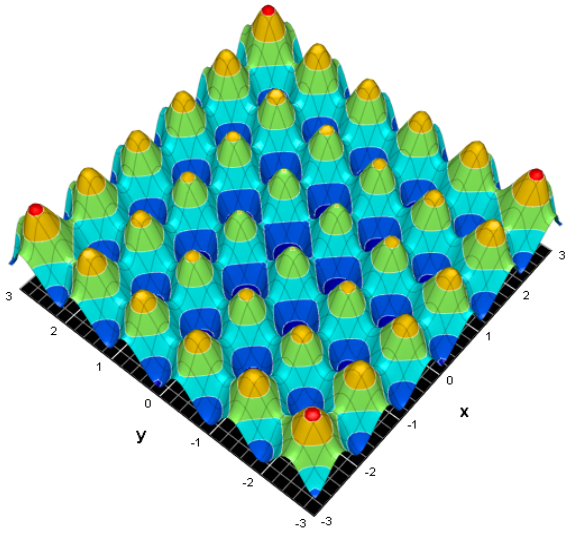


Figura 4.7: Superfície da versão bidimensional da função Rastrigin.

4.4 Configuração dos Experimentos

Foram utilizadas quatro métricas para a realização da comparação da abordagem proposta neste trabalho com o NichePSO, GSO e o dFSS:

- Número de soluções das funções encontradas pelo algoritmo;
- Configuração com menor custo computacional capaz de localizar 95% das soluções corretas;
- Número total de soluções retornadas pelo algoritmo;

- Número de soluções retornadas pelo algoritmo que não correspondem a nenhuma solução da função teste.

Porém, nas duas últimas métricas não foi possível realizar comparações com o GSO pois, segundo Madeiro [3], o GSO não oferece como retorno os pontos ótimos encontrados.

Nos experimentos, o desempenho de todos os algoritmos foi analisado variando o tamanho da população e o número de iterações. Os valores utilizados para o tamanho da população estão descritos na Tabela 4.5, sendo estes dependentes de qual função teste será utilizada. Para o número de iterações, os valores são os mesmos para todas as funções. Porém, estes devem ser diferentes de acordo com a técnica utilizada. Para o NichePSO e o GSO, os valores utilizados para o número de iterações são {50,100,150,...,500}. Porém, o FSS original, base do dFSS e da nova abordagem, executa duas chamadas à função *fitness* a cada iteração. Para que as comparações sejam feitas de forma justa, o número de iterações para a execução do dFSS e da nova abordagem deve ser a metade do utilizado nos outros dois métodos, sendo tais valores representados pelo conjunto {25,50,75,...,250}.

Para que o desempenho dos algoritmos seja avaliado, 30 simulações de cada algoritmo (em cada função) são executadas para cada combinação possível na relação população em função do número de iterações. Por fim, para cada combinação, são calculadas as médias das quatro métricas. Na seção seguinte, os resultados das simulações são apresentados e discutidos.

Tabela 4.5: Valores utilizados nos experimentos para o tamanho da população.

Funções	Valores
Equal Peaks A, Equal Peaks B, Himmelblau, Peaks e Random Peaks	5, 10, 15, ..., 200, 210, 220, ..., 350
Griewank	5, 10, 25, 50, 100, 150, 200, 250, ..., 1300, 1400
Rastrigin	5, 10, 25, 50, 100, 150, 200, 250, ..., 1000

5 *Resultados dos Experimentos*

Este capítulo apreseta e discute todos os resultados obtidos através das simulações descritas no Capítulo 4. Cada seção apresenta os testes realizados para cada função, sendo realizado, no fim, um resumo dos experimentos.

A partir deste ponto, a nova abordagem será referenciada pela sigla *FSSm* (FSS modificado). Neste capítulo, é utilizado o termo “simulação” para cada vez que o algoritmo for executado. Além disso, o termo “experimento” será utilizado para designar 30 simulações com uma mesma configuração.

5.1 Equal Peaks A

Como domínio da função Equal Peaks A, foi utilizado o intervalo $[-\pi,\pi]$. O valor atribuido ao parâmetro m foi 2. Dessa forma, Equal Peaks A possui 9 soluções a serem encontradas.

Para o NichePSO, é possível observar na Figura 5.1a que é preciso pelo menos 100 chamadas à função *fitness* (visto que ocorre 1 chamada por iteração) e 95 indivíduos para que o algoritmo seja capaz de localizar pelo menos 95% das soluções da função. A técnica foi capaz de encontrar, na média das 30 simulações, tal percentual de todas as soluções da função em 68.18% dos experimentos realizados. De acordo com a Figura 5.2a, o algoritmo retorna, no total, 54.73 soluções, calculando-se a média das 30 simulações executada em cada experimento (em cada combinação possível de número de indivíduos e número de iterações). Como pode ser observado na Figura 5.3a, dessas soluções, em média, 47.19 destas, ou seja, 86.27%, são soluções que não correspondem às soluções da função Equal Peaks A.

Para o GSO, observando a Figura 5.1b, nota-se que é necessário pelo menos 100 chamadas à função *fitness* e 80 indivíduos para que o aloritmo seja capaz de localizar pelo menos 95% das soluções da função. Calculando-se a média das 30

simulações para cada experimento (cada combinação entre número de indivíduos e número de iterações), em 72.55% dos casos o GSO foi capaz de localizar pelo menos 95% das soluções da função Equal Peaks A. Como dito na Seção 4.4, não foi possível contabilizar o número total de soluções retornadas pelo GSO, tampouco o número de soluções que não correspondem às soluções da função em questão.

Como pode ser observado na Figura 5.1c, o dFSS conseguiu encontrar 95% das soluções de Equal Peaks A em 74% dos experimentos. Porém, devido às duas chamadas à função *fitness* feitas a cada iteração pelo dFSS, foram necessárias, pelo menos, 100 chamadas à função (50 iterações) e 130 indivíduos para encontrar pelo menos 95% das soluções, o que apresenta um custo um pouco maior para obter tal resultado, se comparado com o NichePSO e o GSO. No entanto, como pode ser observado na Figura 5.2b o dFSS obteve uma média de soluções retornadas muito mais próxima do número de soluções de Equal Peaks A do que o NichePSO: 8.71. Dessas soluções retornadas, de acordo com a Figura 5.3b em média, 0.43 não correspondem às soluções corretas, o que equivale a 4.94% do valor da média do total de soluções.

A nova abordagem encontrou, em 82.18% dos experimentos realizados, 95% das soluções de Equal Peaks A, superando as três técnicas anteriores. Além disso, segundo a Figura 5.1d, é necessário pelo menos 100 chamadas à função *fitness* e apenas 40 indivíduos para encontrar 95% das soluções da função, o que representa um menor custo para obter tais resultados, se comparado com as outras técnicas. No entanto, segundo a Figura 5.2c, a quantidade média de soluções retornadas é igual a 25.5, superando apenas o NichePSO. De acordo com a Figura 5.3c, nota-se que número médio de soluções não-ótimas é igual a 15.2, o que representa 56.61% da média do total de soluções retornadas pela nova abordagem.

5.2 Equal Peaks B

Como domínio da função Equal Peaks B, foi utilizado o intervalo $[-5,5]$. Dessa forma, Equal Peaks B possui 12 soluções a serem encontradas.

Para o NichePSO, é possível observar na Figura 5.4a que é preciso pelo menos 100 chamadas à função *fitness* e 210 indivíduos para que o algoritmo seja capaz de localizar pelo menos 95% das soluções da função. Em 34.55% dos experimentos, o NichePSO foi capaz de encontrar pelo menos 95% do total de soluções possíveis da função Equal Peaks B. De acordo com a Figura 5.5a, o algoritmo retorna, no to-

tal, 54.42 soluções, calculando-se a média das 30 simulações executada em cada experimento. Como pode ser observado na Figura 5.6a, o NichePSO retornou, em média, 45.01 soluções não-ótimas, o que corresponde a 82.71% da média do total de soluções retornadas pelo algoritmo.

Para o GSO, observando a Figura 5.4b, nota-se que é necessário de, pelo menos, 150 chamadas à função *fitness* e 125 indivíduos para que o aloritmo seja capaz de localizar pelo menos 95% das soluções da função. Em 52% dos experimentos, o GSO foi capaz de encontrar pelo menos 95% do total de soluções possíveis da função Equal Peaks B. Como já foi explicado na seção anterior, não foi possível contabilizar o número total de soluções retornadas pelo GSO, tampouco o número de soluções que não correspondem às soluções da função em questão.

Segundo a Figura 5.4c, o dFSS conseguiu encontrar 95% das soluções da função em 64.55% dos experimentos. Para que sejam encontradas 95% das soluções da função Equal Peaks B, foram necesárias pelo menos 150 chamadas à função *fitness* e 110 indivíduos, o que representa, dessa vez, um custo menor para a obtenção de tais resultados, se comparado com o NichePSO e o GSO. Além disso, como pode ser observado na Figura 5.5b o dFSS obteve uma média de soluções retornadas muito mais próxima do número de soluções de Equal Peaks B do que o NichePSO: 12.31. De acordo com a Figura 5.6b, em média, 1.68 soluções não correspondem às soluções da função, o que representa 13.65% das total de soluções retornadas pelo dFSS.

Superando as demas técnicas, em 82.18% dos experimentos realizados, 95% das soluções de Equal Peaks B foram encontradas pela nova abordagem. Além disso, segundo a Figura 5.4d, é necessário pelo menos 100 chamadas à função *fitness* e apenas 50 indivíduos para encontrar 95% das soluções da função, o que representa o menor custo para a obtenção de tal resultado entre as três técnicas. No entanto, segundo a Figura 5.5c, o valor médio do total de soluções retornadas pela nova abordagem é igual a 40.3, superando, mais uma vez, apenas o NichePSO. De acordo com a Figura 5.6c, nota-se que número médio de soluções não-ótimas é igual a 19.5, o que representa 48.39% da média do total de soluções retornadas pelo algoritmo.

5.3 Griewank

Como domínio da função Griewank, foi utilizado o intervalo $[-29,29]$. Dessa forma, Griewank possui 124 soluções a serem encontradas.

Para o NichePSO, observando a Figura 5.7a, nota-se que é preciso de pelo menos 100 chamadas à função *fitness* e 1400 indivíduos para que o algoritmo seja capaz de localizar pelo menos 95% das soluções da função. Em apenas 0.67% dos experimentos, o NichePSO foi capaz de encontrar pelo menos 95% das soluções da função Griewank. De acordo com a Figura 5.8a, o algoritmo retornou, no total, 233.38 soluções, calculando-se a média das 30 simulações executada em cada experimento. Como pode ser observado na Figura 5.9a, o número médio de soluções retornadas que não correspondem a nenhuma solução da função Griewank foi igual a 6.25, o que corresponde a 2.68% da média do total de soluções retornadas pelo algoritmo, o que significa que as partículas tenderam a convergir para poucas soluções, não executando um busca por outras soluções de forma mais ampla.

Para o GSO, observando a Figura 5.7b, nota-se que é necessário de, pelo menos, 150 chamadas à função *fitness* e 1150 indivíduos para que o aloritmo possa localizar pelo menos 95% das soluções da função. O algoritmo localizou este percentual do número de soluções da função Griewank em 9.33% dos experimentos.

Segundo a Figura 5.7c, o dFSS conseguiu encontrar 95% das soluções da função em 34.67% dos experimentos. Para que fossem encontradas 95% das soluções da função Griewank, foram necesárias pelo menos 400 chamadas à função *fitness* e 600 indivíduos. Além disso, como pode ser observado na Figura 5.8b, o número médio do total de soluções retornadas pelo dFSS foi igual 95.18. De acordo com a Figura 5.9b, foram retornadas, em média, 0.77 soluções que não correspondem às soluções da função Griewank, ou seja, 0.81% do total de soluções retornadas pela técnica.

Em 50% dos experimentos realizados, 95% das soluções da função Griewank foram encontradas pela nova abordagem, sendo este valor superior às outras técnicas. Além disso, segundo a Figura 5.7d, é necessário pelo menos 150 chamadas à função *fitness* e 550 indivíduos para encontrar 95% das soluções da função, o que representa, mais uma vez, o menor custo para a obtenção de tal resultado entre as três técnicas. Porém, segundo a Figura 5.8c, a média do total de soluções retornadas pela nova abordagem é muito alta: 262.3. De acordo com a Figura 5.9c, a média do número de soluções não-ótimas retornadas pelo algoritmo corresponde 32.29% da média do total de soluções retornadas, ou seja, 95.2 soluções não-ótimas.

5.4 Himmelblau

Como domínio da função Himmelblau, foi utilizado o intervalo $[-6,6]$. Dessa forma, Himmelblau possui 4 soluções a serem encontradas.

Segundo a Figura 5.10a, é necessário pelo menos 100 chamadas de função *fitness* e 60 indivíduos para que o NichePSO encontre mais de 95% das soluções da função em questão. Em 80.09% dos experimentos, o algoritmo foi capaz de encontrar este percentual das soluções da função Himmelblau. De acordo com a Figura 5.11a, o algoritmo retornou, calculando-se a média do total de soluções retornada das 30 simulações executada em cada experimento, 58.52 soluções. Como pode ser observado na Figura 5.12a, o número médio de soluções retornadas que não correspondem a nenhuma solução da função Himmelblau foi igual a 57.65, o que corresponde a 98.51%, o que demonstrou uma dificuldade do algoritmo em convergir para as soluções desta função, mesmo obtendo um bom percentual das soluções corretas retornadas.

Para o GSO, observando a Figura 5.10b, nota-se que é necessário de, pelo menos, 100 chamadas à função *fitness* e 110 indivíduos para que o aloritmo possa localizar pelo menos 95% das soluções da função. O algoritmo localizou tal percentual do número de soluções da função Himmelblau em 73.09% dos experimentos.

Segundo a Figura 5.10c, o dFSS conseguiu encontrar 95% das soluções da função em 86.91% dos experimentos. Para que fossem encontradas 95% das soluções desta função, foram necesárias pelo menos 100 chamadas à função *fitness* e 75 indivíduos. Além disso, como pode ser observado na Figura 5.11b, o dFSS retornou 4.59 soluções, em média, durante os experimentos. De acordo com a Figura 5.12b, foram retornadas, em média, 0.75 soluções que não correspondem às soluções da função Himmelblau, o que corresponde a 16.34% da média do total de soluções retornadas pela técnica.

A nova abordagem foi capaz de encontrar, em 88.36% dos experimentos, mais de 95% das soluções da função em questão. Além disso, segundo a Figura 5.10d, é necessário pelo menos 100 chamadas à função *fitness* e 45 indivíduos para encontrar 95% das soluções da função, representando o menor custo para obtenção de tal resultado entre as técnicas usadas para comparação. No entanto, segundo a Figura 5.11c, a média do total de soluções retornadas pela nova abordagem é 41. De acordo com a Figura 5.12c, a média do número de soluções não-ótimas retornadas pelo algoritmo corresponde 77.56% da média do total de soluções retornadas, ou seja, 31.8 soluções

que não correspondem com as soluções da função Himmelblau, o que demonstra, assim como aconteceu com o algoritmo NichePSO, uma dificuldade de convergência do algoritmo para as soluções desta função.

5.5 Peaks

Como domínio da função Peaks, foi utilizado o intervalo $[-3,3]$. Dessa forma, Peaks possui 3 soluções a serem encontradas.

Segundo a Figura 5.13a, é necessário pelo menos 50 chamadas de função *fitness* e 70 indivíduos para que o NichePSO encontre mais de 95% das soluções da função em questão, sendo este o menor custo para obtenção de tal resultado. Em 82.73% dos experimentos, o algoritmo foi capaz de encontrar este percentual das soluções da função Peaks. Segundo a Figura 5.14a, a técnica retornou, em média, 62.05 soluções. Como pode ser observado na Figura 5.15a, o número médio de soluções retornadas que não correspondem a nenhuma solução da função Himmelblau foi igual a 60.5, o que corresponde a 97.5%, demonstrando, assim como ocorreu com a função Himmelblau, uma dificuldade de convergência dos indivíduos para as soluções da função em questão.

Segundo a Figura 5.13b, nota-se a necessidade de, pelo menos, 100 chamadas à função *fitness* e 80 indivíduos para que o aloritmo possa localizar pelo menos 95% das soluções da função. Em 78.18% dos experimentos, o algoritmo GSO foi capaz de encontrar tal percentual das soluções de Peaks.

Segundo a Figura 5.13c, o dFSS obteve 95% das soluções da função em 83.45% dos experimentos, sendo este o maior percentual alcançado entre as técnicas aplicadas á função Peaks. Para que fossem encontradas 95% das soluções desta função, foram necesárias pelo menos 100 chamadas à função *fitness* e 90 indivíduos. Além disso, como pode ser observado na Figura 5.14b, o dFSS retornou 6.31 soluções, em média, durante os experimentos. De acordo com a Figura 5.15b, foram retornadas, em média, 3.5 soluções que não correspondem às soluções da função Peaks, o que corresponde a 55.47% da média do total de soluções retornadas pela técnica.

Em 82.54% dos experimentos, mais de 95% das soluções da função em questão foram encontradas pela nova abordagem. Além disso, segundo a Figura 5.13d, é necessário pelo menos 100 chamadas à função *fitness* e 25 indivíduos para encontrar

95% das soluções da função. Segundo a Figura 5.14c, a nova abordagem retornou, em média, 32.2 soluções. De acordo com a Figura 5.15c, a média do número de soluções não-ótimas retornadas pelo algoritmo corresponde 87.27% da média do total de soluções retornadas, ou seja, 28.1 soluções que não correspondem com as soluções da função Peaks, o que demonstra, assim como aconteceu na função Himmelblau, uma dificuldade de convergência do algoritmo para as soluções da mesma.

5.6 Random Peaks

Como domínio da função Random Peaks, foi utilizado o intervalo $[-5,5]$. Nestes experimentos, $Q = 10$, ou seja, existiram 10 soluções a serem encontradas.

Como é possível ser observado na Figura 5.16a, o NichePSO não foi capaz de encontrar mais de 95% das soluções da função Random Peaks em nenhum dos experimentos realizados. Porém, em 42% dos experimentos, o algoritmo foi capaz de encontrar entre 57% e 66.5% das soluções da função, entre 66.5% e 76% em 31.82% e acima de 76% em nenhum dos experimentos. Segundo a Figura 5.17a, a técnica retornou, em média, 61.64 soluções. Como pode ser observado na Figura 5.18a, o algoritmo retornou, em média 58.6 soluções não-ótimas, o que corresponde a 95.07% da média do total de soluções retornadas.

Segundo a Figura 5.16b, assim como ocorreu com o NichePSO, o GSO não obteve nenhum experimento cuja quantidade de soluções retornadas era maior do que 95% das soluções da função Random Peaks. No entanto, em 32.91% dos experimentos, o algoritmo foi capaz de encontrar entre 57% e 66.5% das soluções da função, entre 66.5% e 76% em 16.36% e acima de 76% em apenas 0.18% dos experimentos.

Segundo a Figura 5.16c, o dFSS também não conseguiu encontrar mais de 95% das soluções da função Random Peaks. Porém, em 38.82% dos experimentos, o algoritmo foi capaz de encontrar entre 57% e 66.5% das soluções da função, entre 66.5% e 76% em 38.55% e acima de 76% em apenas 0.18% dos experimentos. Como pode ser observado na Figura 5.17b, o dFSS retornou 8.13 soluções, em média, durante os experimentos. De acordo com a Figura 5.18b, foram retornadas, em média, 1.9 soluções que não correspondem às soluções da função Peaks, o que corresponde a 19% da média do total de soluções retornadas pela técnica.

De acordo com a Figura 5.16d, a nova abordagem também não foi capaz de encon-

trar mais de 95% das soluções da função. No entanto, em 29.45% dos experimentos, o algoritmo foi capaz de encontrar entre 57% e 66.5% das soluções da função, entre 66.5% e 76% em 59.09% e acima de 76% em 3.09% dos experimentos, obtendo mais experimentos com retorno acima de 57% das soluções da função do que as outras técnicas (91.63%). Segundo a Figura 5.17c, a nova abordagem retornou, em média, 35.7 soluções. De acordo com a Figura 5.18c, a média do número de soluções não-ótimas retornadas pelo algoritmo corresponde 65.83% da média do total de soluções retornadas, ou seja, 23.5 soluções que não correspondem com as soluções da função Random Peaks.

5.7 Rastrigin

Como domínio da função Rastrigin, foi utilizado o intervalo $[-5,5]$. Dessa forma, a função Rastrigin possui 100 soluções a serem encontradas.

Para o NichePSO, é possível observar na Figura 5.19a que a técnica não conseguiu encontrar mais de 95% das soluções da função Rastrigin. De acordo com a Figura 5.20a, o algoritmo retorna, no total, 170.66 soluções, calculando-se a média das 30 simulações executada em cada experimento. Como pode ser observado na Figura 5.21a, dessas soluções, em média, 6.59 destas, ou seja, 3.86%, são soluções que não correspondem às soluções da função Rastrigin.

Assim como ocorreu com o NichePSO, para o GSO, observando a Figura 5.19b, nota-se que o algoritmo não foi capaz de localizar mais de 95% das soluções da função Rastrigin.

Como pode ser observado na Figura 5.19c, o dFSS conseguiu encontrar 95% das soluções de Rastrigin em 23.48% dos experimentos. Porém, foram necessárias, pelo menos, 400 chamadas à função *fitness* e 500 indivíduos para encontrar pelo menos 95% das soluções. No entanto, como pode ser observado na Figura 5.20b o dFSS obteve uma média de 74.26 soluções retornadas. Dessas soluções retornadas, de acordo com a Figura 5.21b, em média, 0.81 não correspondem às soluções corretas, o que equivale a 1.09% do valor da média do total de soluções.

A nova abordagem encontrou, em 56.52% dos experimentos realizados, 95% das soluções de Rastrigin, superando as três técnicas anteriores. Além disso, segundo a Figura 5.19d, é necessário pelo menos 150 chamadas à função *fitness* e 400 indiví-

duos para encontrar 95% das soluções da função, o que representa um menor custo para obter tais resultados, se comparado com o dFSS. No entanto, segundo a Figura 5.20c, a quantidade média de soluções retornadas é igual a 153.5. De acordo com a Figura 5.21c, nota-se que número médio de soluções não-ótimas é igual a 37.3, o que representa 24.3% da média do total de soluções retornadas pela nova abordagem.

5.8 Resumo dos Resultados dos Experimentos

Para uma melhor e mais rápida visualização dos resultados, nesta seção será apresentado um resumo dos mesmos. Como pode ser observado nas Tabelas 5.1 e 5.2, a nova abordagem superou todas as técnicas em 6 das 7 funções utilizadas para teste, encontrando mais de 95% das soluções das funções em um maior número de experimentos, perdendo apenas na função Peaks, na qual, mesmo assim, obteve um desempenho muito próximo da técnica melhor sucedida. Além disso, segundo a Figura 5.3, com a nova abordagem é possível encontrar 95% das soluções utilizando uma configuração menos custosa em 6 das 7 funções. Nesta tabela, o conteúdo de cada célula é representado por $x/y(z)$, onde x é o número de chamadas à função *fitness*, y é o número de indivíduos, ambos valores utilizados como parâmetros da configuração do algoritmo em cada experimento, e $z = \sqrt{x^2 + y^2}$, ou seja, z é a distância euclidiana entre tal configuração representada como um vetor bidimensional $[x, y]$ e a configuração com menor custo possivel, ou seja, $[0, 0]$, onde seria utilizado 0 chamadas à função e 0 indivíduos. Obviamente, tal configuração não é factível.

Quanto ao número de soluções retornadas pelo algoritmo e o número de soluções não-ótimas, a nova abordagem obtem, na maior parte das funções, um resultado intermediário entre o NichePSO e o dFSS, sendo esta última a técnica com melhores resultados neste quesito, em todas as funções. O fraco desempenho da nova abordagem neste requisito pode ser interpretado como um indicativo da baixa capacidade do algoritmo em convergir os indivíduos nas soluções corretas, retornando um bom número de soluções corretas, porém, mantendo outros indivíduos em locais que não correspondem com tais soluções. Tais resultados são exibidos, resumidamente, nas Tabelas 5.4 e 5.5. Na Tabela 5.4, os valores exibidos em cada célula corresponde ao módulo da diferença entre o número das soluções corretas de cada função e a média das soluções retornadas em cada experimento.

Observa-se que a nova abordagem obteve um maior número de soluções encon-

tradas do que as outras técnicas na maioria das funções (6 de 7). Além disso, com uma configuração menos custosa, ou seja, menos indivíduos e menos iterações, o FSSm já é capaz de obter bons resultados (retornar mais de 95% das soluções da função objetivo). No entanto, a técnica retorna muitas soluções, se comparada com o dFSS. Além disso, mais uma vez, comparando com o dFSS, o FSSm retorna um alto percentual de soluções não-ótimas neste grupo de soluções retornadas. Em uma situação de uso real, o agente decisor usuário do algoritmo teria como resposta um grande número de soluções possíveis, porém, devendo escolher manualmente algumas poucas destas. Apesar da nova abordagem apresentar estas desvantagens em relação ao dFSS, o FSSm supera o NichePSO e o GSO em praticamente todos os critérios e funções.

Tabela 5.1: Desempenho dos algoritmos nas sete funções.

	NichePSO	GSO	dFSS	FSSm
Equal Peaks A	68.18%	72.55%	74%	**82.18%**
Equal Peaks B	34.55%	52%	64.55%	**82.18%**
Griewank	0.67%	9.33%	34.67%	**50%**
Himmelblau	80.09%	73.09%	86.91%	**88.36%**
Peaks	82.73%	78.18%	**83.45%**	82.54%
Random Peaks	0%	0%	0%	0%
Rastrigin	0%	0%	23.48%	**56.52%**

Tabela 5.2: Desempenho dos algoritmos na função Random Peaks.

	NichePSO	GSO	dFSS	FSSm
>= 7.6	0%	0.18%	0.18%	**3.09%**
>= 6.65 e < 7.6	31.82%	16.36%	38.55%	**59.09%**
>= 5.7 and < 6.65	42%	32.91%	38.82%	**29.45%**
Total >= 5.7	73.82%	49.45%	77.55%	**91.63%**

Tabela 5.3: Configuração com menor custo para obtenção de pelo menos 95% das soluções das funções.

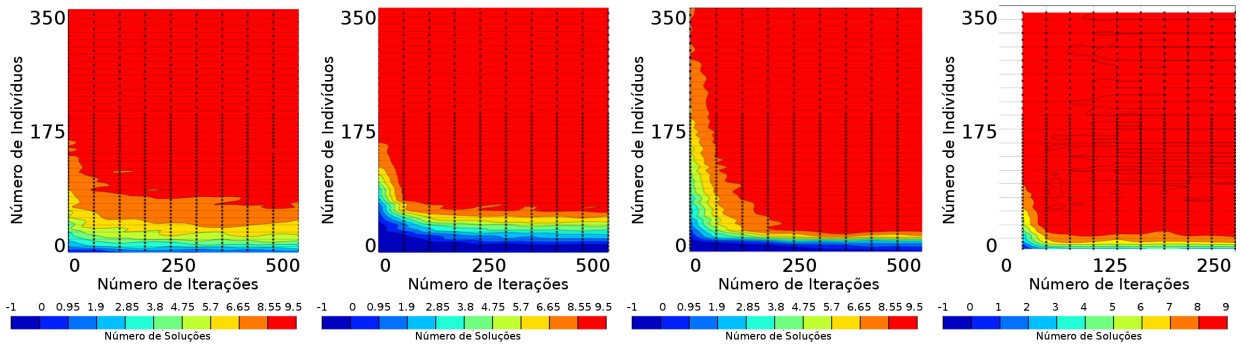
	NichePSO	GSO	dFSS	FSSm
Equal Peaks A	100/95 (137.93)	100/80 (128.06)	100/130 (164.01)	**100/40 (107.70)**
Equal Peaks B	100/210 (232.59)	150/125 (195.25)	150/110 (186.01)	**100/50 (111.80)**
Griewank	100/1400 (1403.57)	150/1150 (1202.08)	400/600 (721.11)	**150/550 (570.09)**
Himmelblau	100/60 (116.62)	100/110 (148.66)	100/75 (125)	**100/45 (109.66)**
Peaks	**50/70 (86.02)**	100/80 (128.06)	100/90 (134.53)	100/25 (103.08)
Random Peaks	-	-	-	-
Rastrigin	-	-	400/500 (640.31)	**150/400 (427.2)**

Tabela 5.4: Módulo da diferença entre o número de soluções encontradas pelos algoritmos e a quantidade de soluções existentes nas funções.

	NichePSO	dFSS	FSSm
Equal Peaks A	45.73	**0.29**	16.5
Equal Peaks B	42.42	**0.31**	28.3
Griewank	109.38	**28.82**	138.3
Himmelblau	54.52	**0.59**	37
Peaks	59.05	**3.31**	29.2
Random Peaks	51.64	**1.87**	25.7
Rastrigin	70.66	**25.74**	53.5

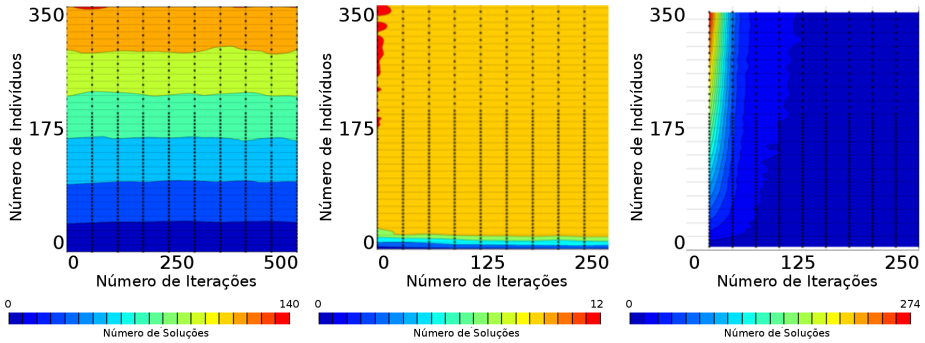
Tabela 5.5: Percentual de soluções que não correspondem às soluções da função em questão em relação ao total de soluções retornadas por cada técnica.

	NichePSO	dFSS	FSSm
Equal Peaks A	86.27%	**4.94%**	27.77%
Equal Peaks B	82.71%	**13.65%**	48.39%
Griewank	2.68%	**0.81%**	36.29%
Himmelblau	98.51%	**16.34%**	77.56%
Peaks	97.50%	**55.47%**	87.81%
Random Peaks	95.07%	**19%**	65.83%
Rastrigin	3.86%	**1.09%**	24.30%



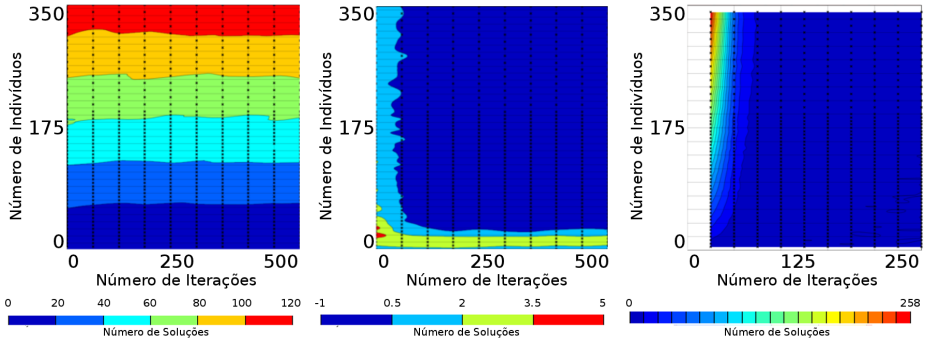
(a) NichePSO (b) GSO (c) dFSS (d) FSSm

Figura 5.1: Soluções de Equal Peaks A



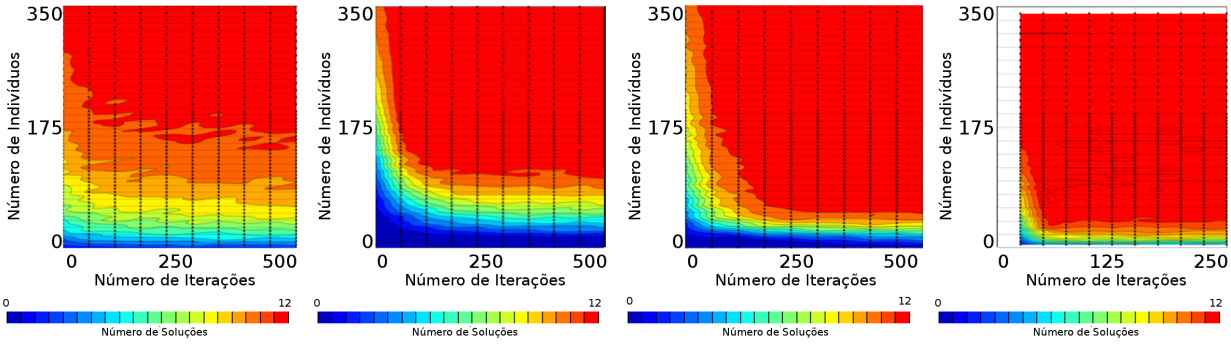
(a) NichePSO (b) dFSS (c) FSSm

Figura 5.2: Total de soluções retornadas por cada técnica (Equal Peaks A).



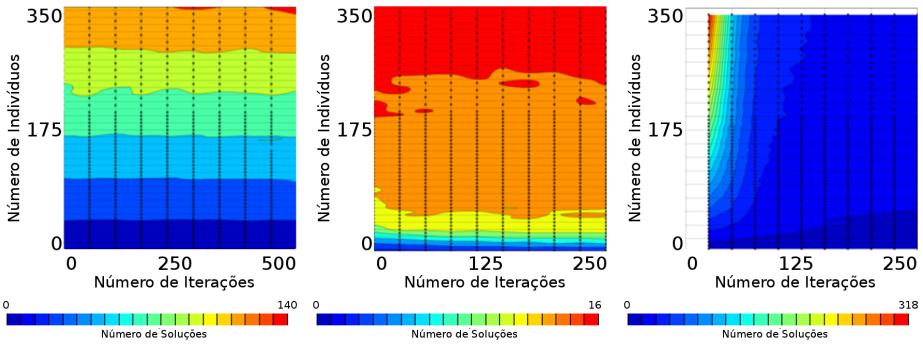
(a) NichePSO (b) dFSS (c) FSSm

Figura 5.3: Número de soluções que não correspondem às soluções de Equal Peaks A.



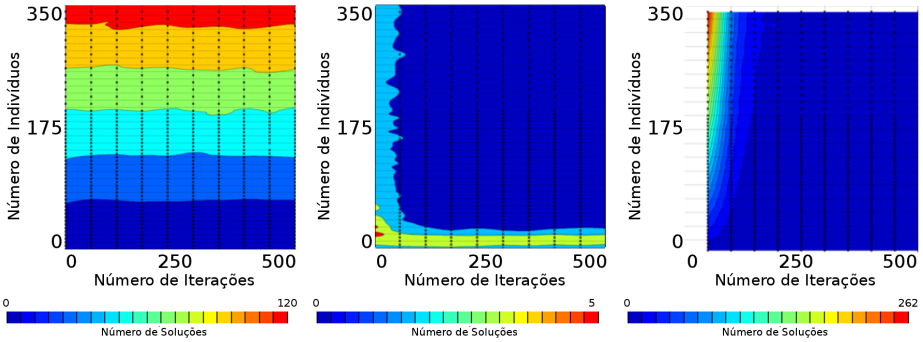
(a) NichePSO (b) GSO (c) dFSS (d) FSSm

Figura 5.4: Soluções de Equal Peaks B.



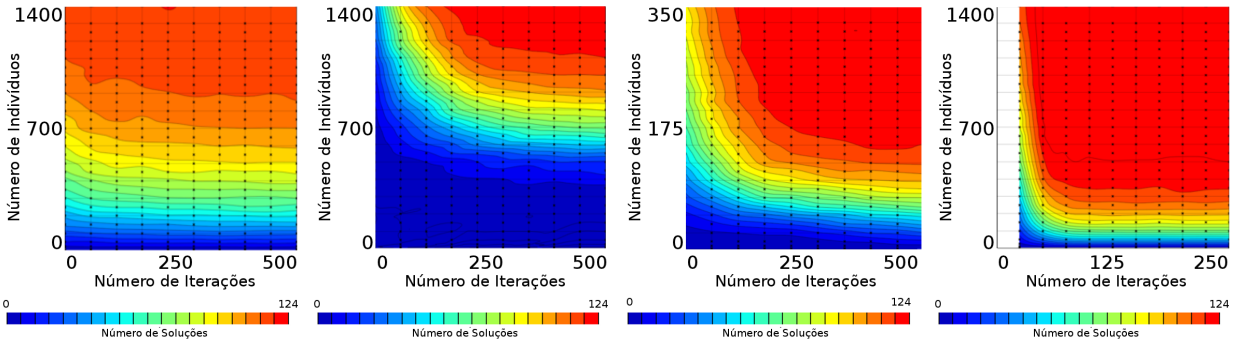
(a) NichePSO (b) dFSS (c) FSSm

Figura 5.5: Total de soluções retornadas por cada técnica (Equal Peaks B).



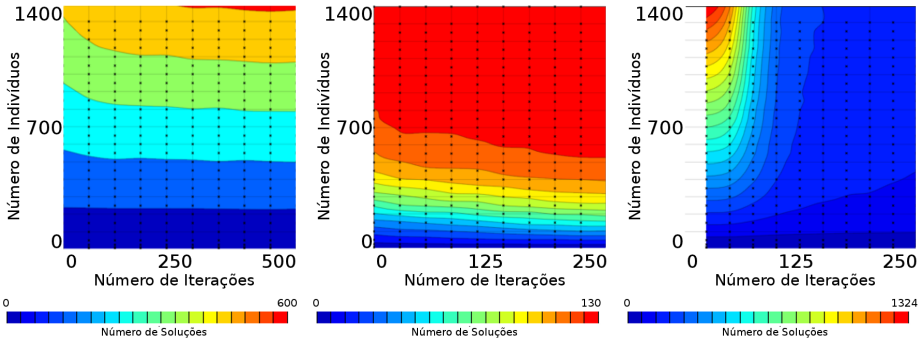
(a) NichePSO (b) dFSS (c) FSSm

Figura 5.6: Número de soluções que não correspondem às soluções de Equal Peaks B.



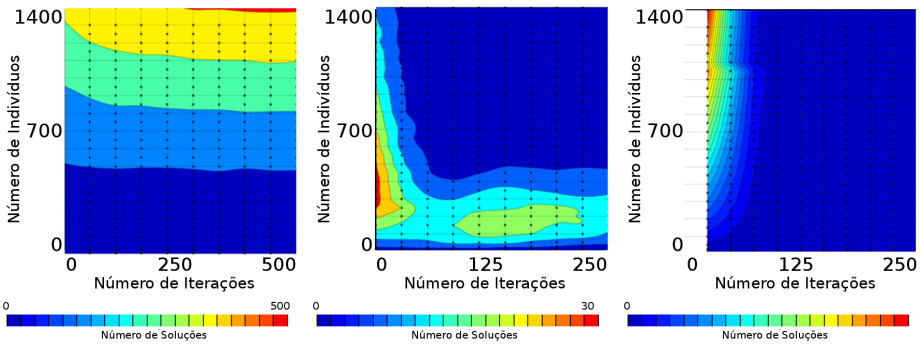
(a) NichePSO (b) GSO (c) dFSS (d) FSSm

Figura 5.7: Soluções de Griewank.



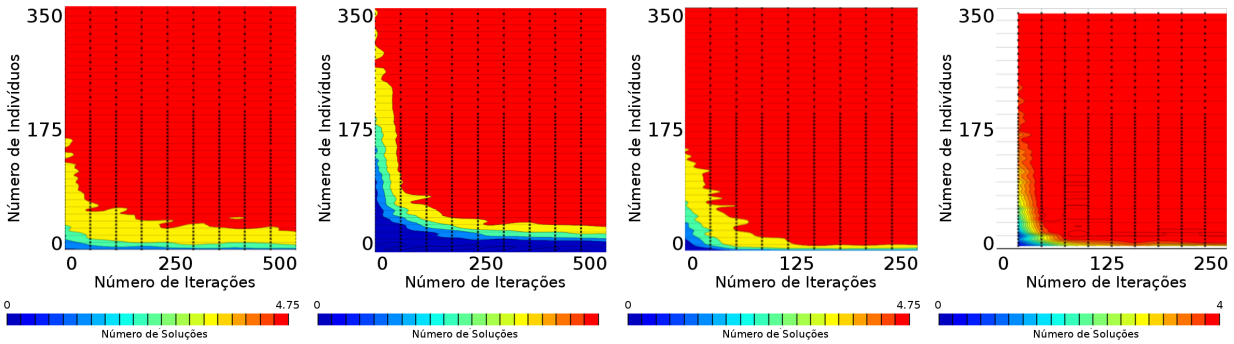
(a) NichePSO (b) dFSS (c) FSSm

Figura 5.8: Total de soluções retornadas por cada técnica (Griewank).



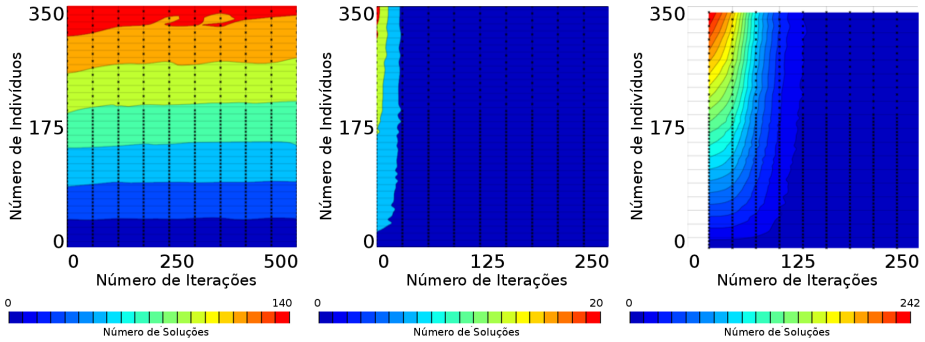
(a) NichePSO (b) dFSS (c) FSSm

Figura 5.9: Número de soluções que não correspondem às soluções de Griewank.



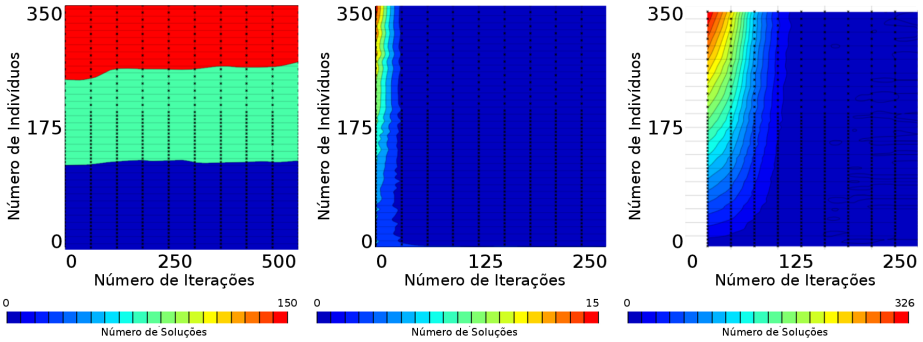
(a) NichePSO (b) GSO (c) dFSS (d) FSSm

Figura 5.10: Soluções de Himmelblau encontradas.



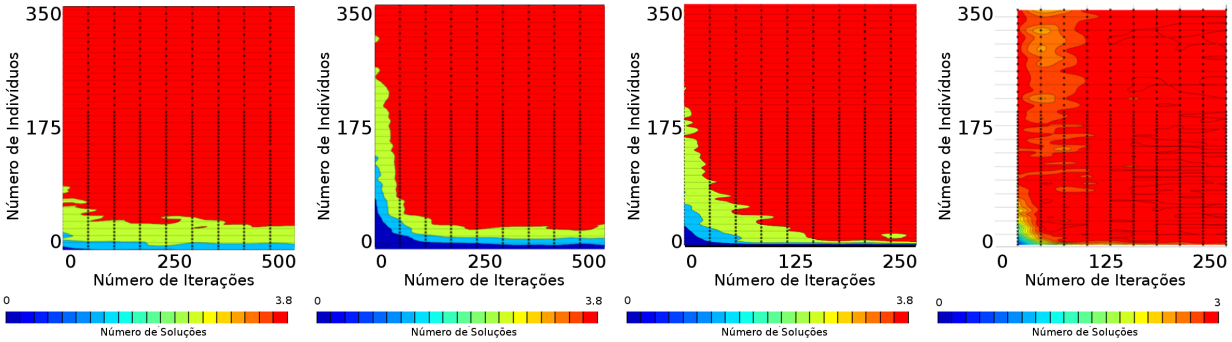
(a) NichePSO (b) dFSS (c) FSSm

Figura 5.11: Total de soluções retornadas por cada técnica (Himmelblau).



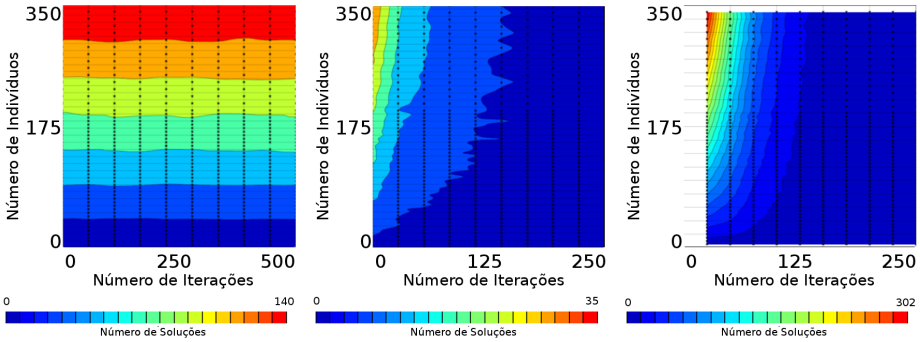
(a) NichePSO (b) dFSS (c) FSSm

Figura 5.12: Número de soluções que não correspondem às soluções de Himmelblau.



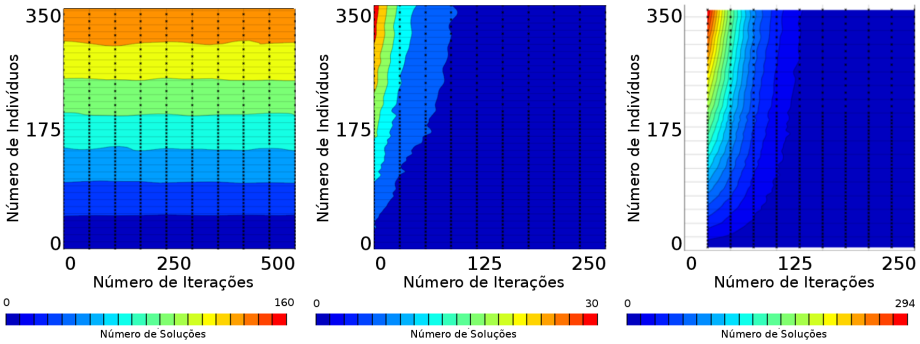
(a) NichePSO (b) GSO (c) dFSS (d) FSSm

Figura 5.13: Soluções de Peaks encontradas.



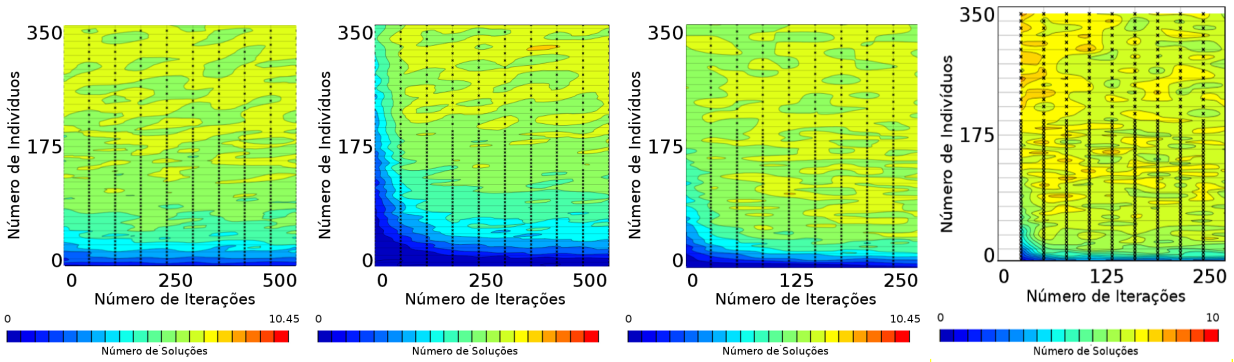
(a) NichePSO (b) dFSS (c) FSSm

Figura 5.14: Total de soluções retornadas por cada técnica (Peaks).



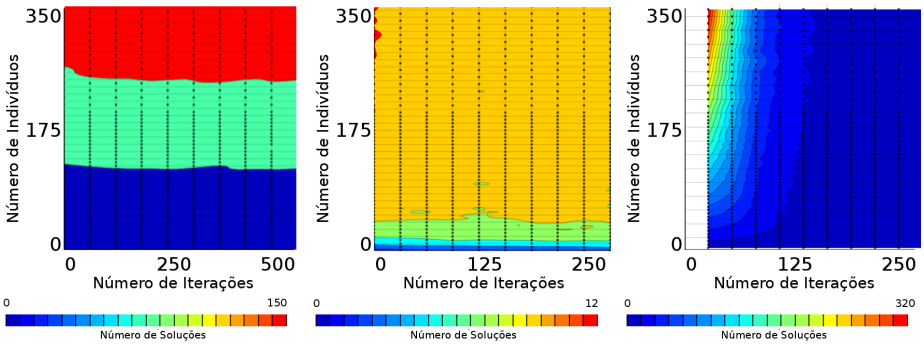
(a) NichePSO (b) dFSS (c) FSSm

Figura 5.15: Número de soluções que não correspondem às soluções de Peaks.



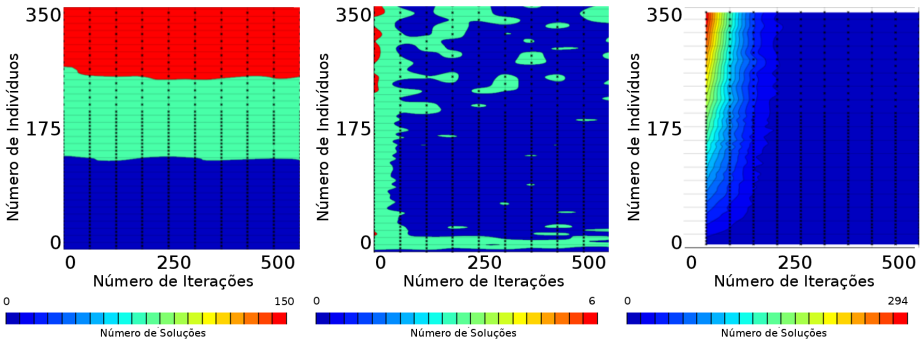
(a) NichePSO (b) GSO (c) dFSS (d) FSSm

Figura 5.16: Soluções de Random Peaks.



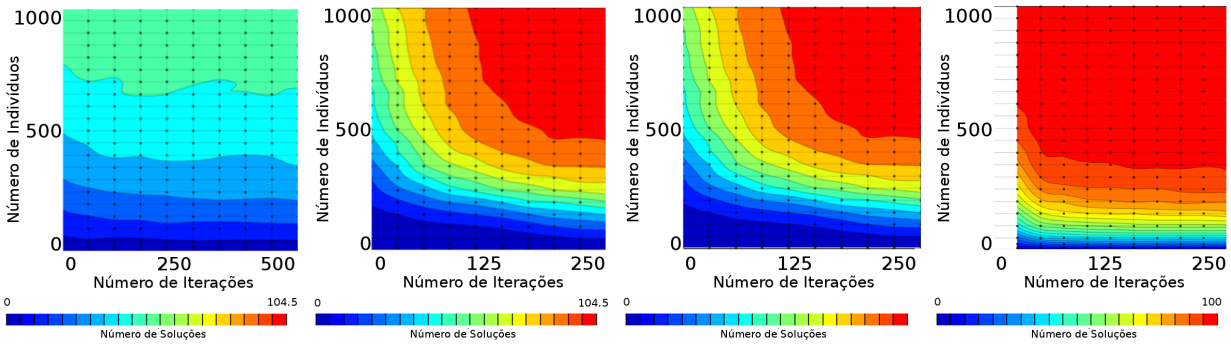
(a) NichePSO (b) dFSS (c) FSSm

Figura 5.17: Total de soluções retornadas por cada técnica (Random Peaks).



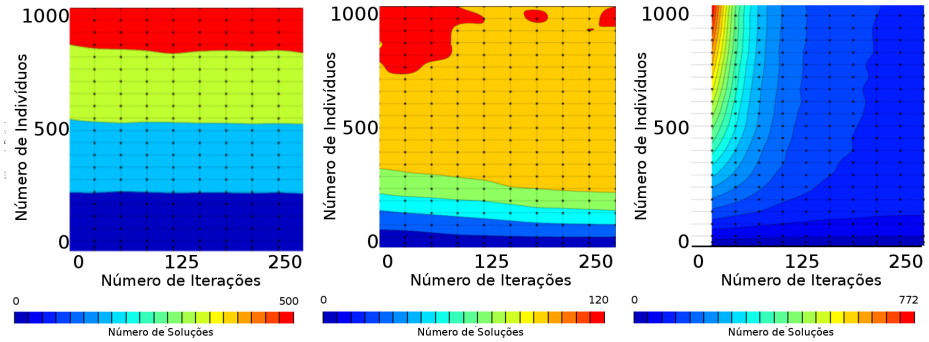
(a) NichePSO (b) dFSS (c) FSSm

Figura 5.18: Número de soluções que não correspondem às soluções de Random Peaks.



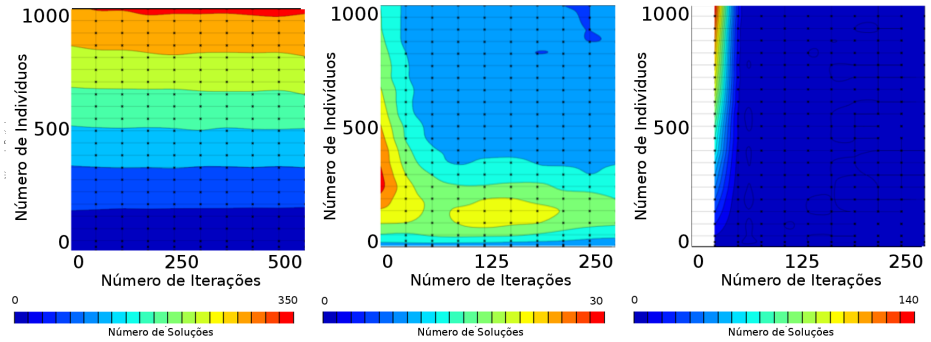
(a) NichePSO (b) GSO (c) dFSS (d) FSSm

Figura 5.19: Soluções de Rastrigin.



(a) NichePSO (b) dFSS (c) FSSm

Figura 5.20: Total de soluções retornadas por cada técnica (Rastrigin).



(a) NichePSO (b) dFSS (c) FSSm

Figura 5.21: Número de soluções que não correspondem às soluções de Rastrigin.

6 Conclusões e Trabalhos Futuros

Neste Trabalho de Conclusão de Curso, foi proposta uma nova heurística para ser aplicada no algoritmo FSS objetivando o embasamento da criação de um novo algoritmo para otimização multisolução de problemas multimodais, fazendo com que os cardumes possuam a capacidade de serem divididos em subcardumes. Observou-se que a nova abordagem é capaz de encontrar um número superior de soluções se comparada com as outras técnicas utilizadas como comparação neste trabalho. Além disso, é possível obter excelentes resultados em uma busca utilizando uma configuração menos custosa do que com as demais técnicas. A desvantagem desta nova abordagem está no número de diferentes soluções retornadas, além do percentual de soluções erradas entre estas. Porém, a nova abordagem ainda superou o NichePSO nesses dois últimos quesitos, perdendo apenas para o dFSS.

Com isso, conclui-se que o uso do peso dos peixes como elemento de segregação populacional pode ser uma melhor alternativa do que as propostas anteriormente realizadas, como a densidade no dFSS. Os resultados aqui apresentados demonstraram uma alta capacidade de detecção de múltipla soluções, devido à habilidade de exploração coordenada do espaço de busca. Além disso, o algoritmo é capaz de encontrar bons resultados com um menor esforço computacional. Outra vantagem é a necessidade de ajuste dos mesmos parâmetros do FSS, pois, ao contrário do que aconteceu no dFSS, nenhum parâmetro extra foi acrescentado. Porém, como desvantagem, a nova abordagem demonstrou uma baixa capacidade de convergência dos subcardumes nas soluções encontradas, o que causa um grande número de soluções retornadas, sendo uma parte considerável destas soluções não-ótimas. Em outras palavras, o algoritmo precisa de mais tempo para convergir os indivíduos nas soluções ótimas da função objetivo.

Como trabalho futuro, utilizando a nova heurística proposta neste trabalho, um novo algoritmo para otimização multisolução de funções multimodais será desenvolvido. Para tal, deve-se criar um mecanismos de aceleração da convergência dos in-

divíduos para as soluções corretas das funções e investigar variações no critério de dominância dos peixes, mas sem perder diversidade na busca, para que seja possível retornar um grande número de diferentes soluções.

Além disso, pretende-se realizar testes precisos de desempenho (custo computacional), além da aplicação de métodos de avaliação anda mais robustos para a validação da abordagem, além de realizar experimentos com funções mais complexas e com um maior número de dimensões.

Referências Bibliográficas

[1] KENNEDY, J.; EBERHART, R.; SHI, Y. *Swarm Intelligence*. 1. ed. [S.l.]: Morgan Kaufmann Publishers, 2001.

[2] ENGELBRECHT, A. P. *Computational Intelligence An Introduction*. 1. ed. [S.l.]: Wiley & Sons, 2007.

[3] MADEIRO, S. *Buscas Multimodais por Cardumes Baseados em Densidade*. Dissertação (Mestrado) — Universidade de Pernambuco, 2009.

[4] BONABEAU, E.; DORIGO, M.; THERAULAZ, G. *Swarm intelligence: from natural to artificial systems*. [S.l.]: Oxford University Press, Inc., 1999.

[5] KENNEDY, J.; EBERHART, R. A new optimizer using particle swarm theory. In: *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. [S.l.: s.n.], 1995. p. 39–43.

[6] DORIGO, M. *Optimization, learning and natural algorithms*. Tese (Doutorado) — Politecnico di Milano, 1992.

[7] FLHO, C. J. A. B. et al. A novel search algorithm based on fish school behavior. In: *IEEE International Conference on Systems, Man, and Cybernetics*. [S.l.: s.n.], 2008.

[8] OZCAN, E.; YILMAZ, M. Particle swarms for multimodal optimization. *Lecture Notes in Computer Science*, v. 4431, p. 366–375, 2007.

[9] LI, L.; HONG-QI, L.; SHAO-LONG, X. Particle swarm multi_optimizer for locating all local solutions. In: *Proceedings of the IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2008. p. 1040–1046.

[10] BRITS, R.; ENGELBRECHT, A. P.; BERGH, F. van der. Locating multiple optima using particle swarm optimization. *Applied Mathematics and Computation*, v. 2, n. 189, p. 1859–1883, 2007.

[11] KRISHNANAND, K.; GHOSE, D. Glowworm swarm optimization for simultaneous capture ofÂ multiple local optima ofÂ multimodal functions. *Swarm Intelligence*, v. 3, n. 2, p. 87–124, 2009.

[12] MADEIRO, S. S. et al. Density as the segregation mechanism in fish school search for multimodal optimization problems. In: *Proceedings of the Second International Conference on Advances in Swarm Intelligence*. [S.l.: s.n.], 2011. v. 2, p. 563–572.

[13] ARORA, J. et al. Global optimization methods for engineering applications: a review. *Structural Optimization*, v. 3-4, n. 9, p. 137–159, 1995.

[14] KAN, A. R.; BOENDER, C.; TIMMER, G. A stochastic approach to global optimization. *Computational Mathematical Programming*, p. 281–308, 1985.

[15] BIEGLER, L. T.; GROSSMANN, I. E. Restrospective on optimization. *Computers and Chemical Engineering*, v. 8, n. 28, p. 1169–1192, 2004.

[16] EBERHART, R. C.; SHI, Y. *Computational Intelligence: Concepts to Implementations*. 1. ed. [S.l.]: Morgan Kaufmann, 2007.

[17] KENNEDY, J. Review of engelbrecht's fundamentals of computational swarm intelligence. *Genetic Programming and Evolvable Machines*, v. 8, n. 1, p. 107–109, 2007.

[18] KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: *Proccedings of the IEEE International Conference on Neural Networks*. [S.l.: s.n.], 1995. v. 4, p. 1942–1948.

[19] SHI, Y.; EBERHART, R. A modified particle swarm optimizer. In: *Proceedings of the IEEE International Conference on Evolutionary Computation*. [S.l.: s.n.], 1998. p. 69–73.

[20] CLERC, M.; KENNEDY, J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, v. 6, p. 58–73, 2002.

[21] BANKS, A.; VINCENT, J.; ANYAKOHA, C. A review of particle swarm optimization. part i: background and development. Kluwer Academic Publishers, Hingham, MA, USA, v. 6, n. 4, p. 467–484, 2007.

[22] BANKS, A.; VINCENT, J.; ANYAKOHA, C. A review of particle swarm optimization. part ii: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. Kluwer Academic Publishers, Hingham, MA, USA, v. 7, n. 1, p. 109–124, 2008.

[23] DORIGO, M.; GAMBARDELLA, L. M. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, p. 53–66, 1997.

[24] MARSH, L.; ONOF, C. Stigmergic epistemology, stigmergic cognition. *Cognitive Systems Research*, 2007.

[25] KARABOGA, D. *An idea based on honey bee swarm for numerical optimization*. [S.l.], 2005.

[26] KARABOGA, D.; BASTURK, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, p. 459–471, 2006.

[27] KARABOGA, D.; BASTURK, B. Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. In: *Proceedings of the 12th international Fuzzy Systems Association world congress on Foundations of Fuzzy Logic and Soft Computing*. Berlin, Heidelberg: Springer-Verlag, 2007. p. 789–798.

[28] NARASIMHAN, H. Parallel artificial bee colony (pabc) algorithm. In: *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*. [S.l.: s.n.], 2009. p. 306–311.

[29] PASSINO, K. M. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, v. 22, n. 3, p. 52–67, 2002.

[30] XIAOLONG, L.; RONGJUN, L.; PING, Y. A bacterial foraging global optimization algorithm based on the particle swarm optimization. In: *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*. [S.l.: s.n.], 2010. v. 2, p. 22–27.

[31] SUPRIYONO, H.; TOKHI, M. O. Bacterial foraging algorithm with adaptable chemotactic step size. In: *Computational Intelligence, Communication Systems and Networks, 2010 International Conference on*. [S.l.: s.n.], 2010. p. 72–77.

[32] CHO, J. H. et al. Bacterial foraging with quorum sensing based optimization algorithm. In: *Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on*. [S.l.: s.n.], 2009. p. 29–34.

[33] HORN, J. *The Nature of Niching: Genetic Algorithms and The Evolution of Optimal, Cooperative Populations*. Tese (Doutorado) — University of Illinois, 1997.

[34] KRONFELD, M.; ZELL, A. Towards scalability in niching methods. In: *Proceedings of the IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2010. p. 1–8.

[35] KENNEDY, J. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In: *Proceedings of IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 1999. p. 1931–1938.

[36] ENGELBRECHT, A.; LOGGERENBERG, L. van. Enhancing the nichepso. In: *Proceedings of IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2007. p. 2297–2302.