UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

Desenvolvimento de método de inteligência artificial baseado no comportamento de enxames do gafanhoto-do-deserto

TIAGO MARTINS RIBEIRO

São Luís – MA, Brasil
Fevereiro, 2017

TIAGO MARTINS RIBEIRO

Desenvolvimento de método de inteligência artificial baseado no comportamento de enxames do gafanhoto-do-deserto

Dissertação de Mestrado submetida à Coordenação do Programa de Pós-Graduação em Engenharia de Eletricidade (PPGEE) da Universidade Federal do Maranhão (UFMA) como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica na área de concentração de Ciência da Computação.

Orientador: Prof. Dr. Vicente Leonardo Paucar Casas

São Luís – MA, Brasil
Fevereiro, 2017

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

Ribeiro, Tiago Martins.
Desenvolvimento de método de inteligência artificial baseado no comportamento de enxames do gafanhoto-do-deserto / Tiago Martins Ribeiro. - 2017.
93 f.

Orientador(a): Vicente Leonardo Paucar Casas.
Dissertação (Mestrado) - Programa de Pós-graduação em Engenharia de Eletricidade/ccet, Universidade Federal do Maranhão, São Luís, 2017.

1. Algoritmos bioinspirados. 2. Enxame de gafanhotos-do-deserto. 3. Inteligência artificial. 4. Inteligência de enxames. 5. Serotonina. I. Paucar Casas, Vicente Leonardo. II. Título.

Desenvolvimento de método de inteligência artificial baseado no comportamento de enxames do gafanhoto-do-deserto

Tiago Martins Ribeiro

Dissertação aprovada em 20 de fevereiro de 2017

Prof. Dr. Vicente Leonardo Paucar Casas, UFMA
(Orientador)

Profa. Dra. Áurea Celeste da Costa Ribeiro, UEMA
(Membro da Banca Examinadora)

Prof. Dr. Denivaldo Cicero Pavão Lopes, UFMA
(Membro da Banca Examinadora)

Dedico a todos que me apoiaram neste trabalho e deram-me oportunidades para chegar até aqui, em especial a meus pais Sebastião Ribeiro Lima e Maria do Socorro Lima Martins Ribeiro e a meu irmão Davi Martins Ribeiro.

Agradecimentos

A Deus por ter me dado saúde e forças para superar as dificuldades.

A meus pais, que ao longo de minha jornada de estudos, deram-me total apoio nos momentos de dificuldades enfrentados longe de casa.

Ao meu orientador de graduação, Raimundo Pereira da Cunha Neto, que abriu a oportunidade do tema deste trabalho e ajudou-me dar continuidade indo ao mestrado.

Ao meu orientador de mestrado, Prof. Dr. Vicente Leonardo Paucar Casas, pela orientação, paciência, suporte, correções, incentivo e direção ao meu objetivo ao longo deste trabalho, que hoje vislumbro um horizonte superior, eivado pela acendrada confiança no mérito e ética dele presente.

Ao meu colega Raimundo Diniz, pela ajuda, paciência, tempo e orientação diante o trabalho aqui realizado.

Aos meus demais colegas de laboratório, Marília, Rafael, Nilson, Felipe, Antônio e Arthur, pela paciência, amizade, caronas, ajudas nos trabalhos e horas livres de diversão prestadas ao longo destes dois anos de estudos. Também, aos colegas Thiago Pinheiro, Gilberto Nunes e Matias Romário pela ajuda auxiliar nas matérias, pelas caronas e as horas livres de diversão.

Em memória ao Prof. Zair Abdelouahab, que me ensinou em vida através de seu exemplo de superação, que somos mais fortes do que imaginamos ser, e que se dedicarmos nosso tempo aos nossos sonhos através de nossas ações, somos capazes de realiza-los. Mesmo que a vida nos diga que não somos capazes ou nos opunha barreiras físicas a este objetivo, se realmente querermos, seremos capazes de chegar ao pódio.

A esta universidade, seu corpo docente, pela direção e administração que me deram ao longo dos estudos até chegar aqui.

A Capes, pelo auxílio financeiro que foi essencial para dar continuidade ao estudo.

E a todos que direta ou indiretamente fizeram parte de minha formação, o meu muito obrigado.

"As consequências de nossos atos são sempre tão complexas, tão diversas, que predizer o futuro é uma tarefa realmente difícil."

Alvo Dumbledore

Resumo

Problemas complexos de otimização vêm sendo estudados ao longo dos anos por pesquisadores que buscam melhores soluções, estes estudos incentivaram o desenvolvimento de vários algoritmos de inteligência artificial, sendo que uma parte deles são métodos bioinspirados, baseados no comportamento de populações. Estes algoritmos têm como objetivo desenvolver técnicas baseadas na natureza em busca de soluções para estes problemas. Neste trabalho um algoritmo baseado no comportamento de enxames de gafanhotos-do-deserto, o *Locust Swarm Optimizer* (LSO), foi introduzido como objetivo. O comportamento do gafanhoto-do-deserto é apresentado destacando a formação de nuvens de ataques causada por uma monoamina neurotransmissora sintetizada, presente no inseto, conhecido por serotonina. Observando este comportamento, foi desenvolvido o LSO. Ele foi comparado com outras conhecidas técnicas de inteligência artificial através de 23 funções *benchmarks* e também, testado em um problema de despacho econômico. Do ponto de vista dos resultados e da facilidade de implementação, pode-se concluir que o algoritmo LSO é bastante competitivo comparado aos métodos atuais existentes.

Palavras-chave: Inteligência artificial, inteligência de enxames, enxame de gafanhotos-do-deserto, serotonina, algoritmos bioinspirados.

Abstract

Complex optimization problems have been studied over the years by researchers seeking better solutions, these studies have encouraged the development of several algorithms of artificial intelligence, and a part of them are bio-inspired methods, based on the behavior of populations. These algorithms target to develop techniques based on nature in search of solutions to these problems. In this work, was introduced as a purpose, an algorithm based on the behavior of locust swarms, the Locust Swarm Optimizer (LSO). The behavior of the desert locust is introduced highlighting the formation of clouds of attacks caused by a synthesized neurotransmitter monoamine, present on the insect, known as serotonin. Observing this behavior, the LSO was developed. It was compared to other known artificial intelligence techniques through 23 benchmark functions and also tested on an power system economical dispatch problem. From the point of view of the results and the ease of implementation, it can be concluded that the LSO algorithm is very competitive as compared to existing methods.

Keywords: Artificial intelligence, swarm intelligence, desert locust swarm, serotonin, bio-inspired algorithms.

Lista de Figuras

Lista de Tabelas

Página

Lista de Abreviaturas

ACO	: *Ant Colony Optimization*
ANN	: *Artificial Neural Network*
CLONALG	: *CLOnal selection ALGorithm*
CS	: *Cuckoo Search*
DDoS	: *Distributed Denial of Service*
EA	: *Evolutionary Algorithm*
EP	: *Evolutionary Programming*
ES	: *Evolutionary Strategies*
GA	: *Genetic Algorithm*
IA	: Inteligência artificial
IS	: *Immune Systems*
LSO	: *Locust Swarm Optimizer*
MATLAB	: *MATrix LABoratory*
PSO	: *Particle Swarm Optimization*
TSP	: *Travelling Salesman Problem*

Sumário

Página

1 Introdução

1.1 Generalidades

Nas últimas décadas, a evolução do processamento de informações na área da computação, é bastante notória. Tudo isto porque a sociedade se adapta às novas tecnologias em menor tempo e espaço. A Inteligência Artificial (IA), em inglês *Artificial Intelligence* (AI), é uma área da Ciência da Computação que trata do desenvolvimento de métodos baseados na inteligência existente na natureza, como por exemplo, dos animais, de forma a implementar programas computacionais que emulam a inteligência natural, auxiliando assim na produção de algoritmos que reduzem a complexidade.

O uso de dispositivos para facilitar o trabalho da mão de obra humana é cada vez mais presente na vida moderna, tornando-se natural nos dias atuais. O progresso tecnológico está convergindo de forma rápida e cada vez mais são menores as diferenças entre coleta, transporte, armazenamento e processamento de informações [1]. Esta conversão se dá graças aos estudos computacionais, a fim de minimizar o tempo e o processamento das informações processadas. A partir desse estudo vem a ideia de métodos inteligentes para otimizar problemas complexos.

Entre os métodos de inteligência artificial, têm-se os mais tradicionais, propostos inicialmente nos anos 60 e 70, tais como redes neurais artificiais (*Artificial Neural Network* - ANN), lógica *fuzzy*, algoritmos genéticos, algoritmos evolutivos, etc. A partir da década de 1980 começaram a ser propostos os métodos de inteligência artificial enfatizando a inteligência de enxames de partículas, tal como otimização por enxame de partículas ou PSO (*Particle Swarm Optimization*), e também métodos que se inspiram no

comportamento inteligente de colônias de insetos, tais como colônias de formigas, colônias de abelhas, dentre outros.

As aplicações dos métodos de inteligência artificial considerando PSO e colônias de insetos são variadas. Uma aplicação bastante utilizada é na área de otimização de funções objetivo, otimização multiobjetivo, otimização de processos, processamento distribuído, etc.

Nesta pesquisa, um novo método de inteligência artificial baseado no comportamento inteligente dos enxames de gafanhotos-do-deserto que em algumas situações podem formar nuvens de ataque será apresentado.

1.2 Formulação do problema

O uso de modelos baseados em comportamentos de colônias de insetos vem proporcionando ao estudo computacional soluções de otimizações de buscas, no qual os algoritmos criados, a partir de estudos biológicos, ajudam a Ciência da Computação a resolver o paradigma da complexidade. Dorigo [2], em sua tese de doutorado abordou sobre o uso de colônias de formigas para otimização de buscas; destaca-se também Yang e Deb [3], que trouxeram em seu estudo o comportamento de espécies de aves cucos, outra técnica que aborda problemas de otimização. O PSO, é um enxame de partículas originado por Kennedy e Eberthart [4] que também tem destaque em soluções de otimização.

Diversos algoritmos existentes tornam-se importantes na otimização de problemas, verifica-se que algoritmos que envolvem colônias possuem técnicas diferentes favorecendo com que o modelo computacional do comportamento do gafanhoto-do-deserto possa ser estudado, assim, algumas questões precisam ser respondidas: Qual o modelo comportamental do gafanhoto-do-deserto? Quais funções do modelo comportamental do

gafanhoto-do-deserto se aplicam melhores? O modelo comportamental do gafanhoto-do-deserto se destaca dos algoritmos de otimização já existentes?

1.3 Objetivos

O objetivo geral é:

- Contribuir com a proposta de um novo método de inteligência artificial para a solução de problemas nas diversas áreas da ciência e tecnologia.

Os objetivos específicos são:

- Desenvolver um método de inteligência artificial baseado no comportamento do enxame de gafanhotos-do-deserto (*Locust Swarm Optimizer* - LSO);
- Estudar o comportamento biológico do gafanhoto-do-deserto;
- Aplicar o método proposto do enxame de gafanhotos-do-deserto para resolver funções de otimização *benchmarks unimodais*, *multimodais,* e *multimodais* de dimensão fixa;
- Comparar o desempenho do método proposto do enxame de gafanhotos-do-deserto com outros métodos de inteligência artificial existentes na literatura.

1.4 Justificativa

O estudo e desenvolvimento de técnicas para solucionar problemas de otimização utilizando colônias de insetos, vem recebendo destaque ao longo dos anos, visto que a literatura revela técnicas utilizando formigas [2], aves cucos [3], partículas de enxames [4] e outras espécies da natureza para soluções de otimização. Baseando-se nestes algoritmos, surge então o estudo do comportamento do gafanhoto-do-deserto.

O gafanhoto-do-deserto se destaca por trabalhar em colônia na formação de enxames (nuvens) de ataques às plantações, onde durante o período de formação do enxame, esta espécie muda de fase através de estímulos que aumentam sua taxa de serotonina. Este trabalho busca contribuir na construção de um novo algoritmo que auxilie na busca de soluções ótimas de problemas complexos através do comportamento do enxame de gafanhotos-do-deserto, que semelhante às formigas usam um neurotransmissor (serotonina) para mudar de estado, e ao contrário das formigas, buscam indivíduos da mesma espécie para encontrar a solução ótima e assim chegar ao alimento.

1.5 Metodologia

Neste trabalho, a metodologia de pesquisa científica consistirá na realização ordenada de várias etapas incluindo atividades previstas no pré-projeto foi adotada.

Inicialmente, o levantamento do estado da arte no tema proposto, incluindo pesquisa documental, para levantamento de informações de livros, teses, artigos, dissertações, periódicos, anais de congressos e websites, foi utilizada para a conclusão desta pesquisa bibliográfica. Como resultado desta etapa foi elaborado um relatório sobre o estado da arte dos métodos de inteligência artificial inspirados na inteligência existente na natureza.

Em seguida, a informação da pesquisa bibliográfica visando o conhecimento mais aprofundado dos principais métodos de inteligência artificial enfatizando os mais tradicionais e os bioinspirados no comportamento de colônias de insetos, foi estudada.

Outra etapa consiste no estudo de linguagens de programação e aplicativos de ciência da computação para a implementação de programas computacionais eficientes de inteligência artificial, inteligência artificial distribuída e otimização.

Estudo dos principais conceitos biológicos de colônias de insetos e priorizando o comportamento inteligente dos gafanhotos-do-deserto quando formam enxames de ataque para a sua sobrevivência, reprodução, etc.

Formulação básica do modelo matemático e do algoritmo do enxame de gafanhotos-do-deserto como um método de inteligência artificial.

Implementação computacional do algoritmo do método IA do enxame de gafanhotos-do-deserto para aplicações básicas de otimização de problemas envolvendo funções *unimodais* e *multimodais* do tipo *benchmark*.

A metodologia de análise estatística foi utilizada para comparar o desempenho do método do gafanhoto-do-deserto com relação a outros métodos de inteligência artificial existentes na literatura científica.

1.6 Estrutura do trabalho

Este trabalho contém cinco capítulos organizados da seguinte maneira.

O Capítulo 1 descreve os assuntos relativos à proposta do trabalho apresentado.

O Capítulo 2 apresenta conceitos relativos ao estudo dos métodos da inteligência artificial inspirados na natureza.

O Capítulo 3 apresenta conceitos relativos ao estudo do método proposto de inteligência artificial do enxame de gafanhotos-do-deserto.

O Capítulo 4 apresenta os resultados e futuras aplicações do método proposto.

O Capítulo 5 contém as conclusões do trabalho.

2 Métodos de inteligência artificial inspirados na natureza

2.1 Introdução

O uso das tecnologias ocorre de uma forma natural nos dias atuais, uma vez que dispositivos e softwares facilitam o trabalho da mão de obra humana, observa-se que dados e informações estão diariamente sendo processados em dispositivos, sejam eles móveis ou não. A busca das melhores soluções para o trabalho desses dados e informações é vista nas áreas de engenharia da computação e ciência da computação.

Problemas complexos de otimização são um grande desafio para a engenharia, isto se dá ao fato desses problemas crescerem seus espaços de busca progressivamente com o tamanho dos problemas, de modo que os métodos tradicionais de otimização não auxiliam na solução adequada. Por isso, ao longo dos anos muitos algoritmos de otimização foram desenvolvidos para resolver esses problemas [5-9].

O comportamento do mundo real serve de inspiração para o comportamento computacional, pois a natureza está servindo de inspiração para a solução de problemas de otimização auxiliando no desenvolvimento de métodos inspirados na natureza [10-14]. Nos próximos tópicos serão abordados alguns exemplos desses métodos de inteligência artificial inspirados na natureza, assim como conceitos sobre a inteligência artificial.

2.2 Inteligência artificial

O homem é considerado um ser inteligente desde sua formação na espécie *Homo sapiens* (homem sábio). Durante vários anos vêm-se tentando compreender como o homem pensa, ou seja, como sua matéria é capaz de compreender, perceber, prever e até manipular o mundo ao seu redor. A área de IA busca ir além dessas percepções humanas, busca construir entidades inteligentes [15].

Nesta seção irão ser abordados definições da IA, sua classificação e os métodos bioinspirados que são trabalhados para otimizar sistemas de informa inteligente.

2.2.1 Definição

A IA é um campo recente nos estudos das ciências e engenharias, visto que os trabalhos relacionados a este campo começaram somente após a Segunda Guerra Mundial, e o nome foi cunhando por volta de 1956 [15]. Atualmente este campo da pesquisa possui variados subcampos, tanto na parte do aprendizado como na percepção ou mesmo em tarefas mais específicas; como demonstrações de teoremas matemáticos, jogos de raciocínio lógico tipo o xadrez, no comportamento do funcionamento de um veículo automotivo em uma estrada ou mesmo no diagnóstico de doenças na medicina. A IA possui uma grande relevância em qualquer tarefa intelectual, ela é considerada um campo universal [15].

Não existe uma definição exclusiva para inteligência artificial, mas existem várias definições que buscam em diversos grupos chegar próximo a esta definição. Basicamente pode-se dizer que IA é um campo de estudo que tenta fazer com que máquinas computacionais pensem como os seres humanos ou que essas máquinas sejam tão inteligentes quanto ao homem. A seguir, na Tabela 2.1, adaptada de [15], são listadas algumas definições de IA levando em consideração quatro grupos de inteligência, o

pensando como um humano, o pensando racionalmente, o agindo como seres humanos e por fim, o agindo racionalmente.

Tabela 2.1- Definições de IA

Pensando como um humano	**Pensando racionalmente**
"O novo e interessante esforço para fazer os computadores pensarem (...) máquinas com mentes, no sentido total e literal. " [16] "[Automatização de] atividades que associamos ao pensamento humano, atividades como a tomada de decisões, a resolução de problemas, o aprendizado..." [17]	"O estudo das faculdades mentais pelo uso de modelos computacionais. " [18] "O estudo das computações que tornam possível perceber, raciocinar e agir. " [19]
Agindo como seres humanos	**Agindo racionalmente**
"A arte de criar máquinas que executam funções que exigem inteligência quando executadas por pessoas. " [20] "O estudo de como os computadores podem fazer tarefas que hoje são melhor desempenhadas pelas pessoas. " [21]	"Inteligência Computacional é o estudo do projeto de agentes inteligentes. " [22] "IA... está relacionada a um desempenho inteligente de artefatos. " [23]

2.2.2 Classificação

A inteligência artificial se divide basicamente em duas grandes áreas, a IA clássica e a IA moderna [15].

A IA clássica se divide em:

- Resolução de problemas;
- Sistemas baseados no conhecimento;
- Sistemas especialistas.

A IA clássica se deu no período de 1956 a 1979, por ser recente o seu estudo não se sabia claramente o que realizar diante a tantos problemas existentes para se resolver, desta

forma a complexidade era um desafio para os pesquisadores da época. Eles buscaram inicialmente compreender a teoria dos autômatos, redes neurais e o estudo da inteligência. McCarthy convenceu Minsky, Claude Shannon e Nathaniel Rochester a ajudá-lo a reunir pesquisadores que tivessem interesse em estudar estes campos de pesquisa, para que assim dessem início a IA propriamente dita [15].

A priori os primeiros resultados tiveram sucesso, porém eram limitados a equações aritméticas voltadas a resoluções de problemas simples, e de forma automática, com conjuntos de regras e métodos de buscas informados ou não informados [15].

Estas soluções viam de forma automatizada, sem inicialmente ter um comportamento inteligente humano por trás, apenas um comportamento com um conjunto de regras que tendiam a mostrar uma solução para o problema buscado. Assim se deu a fase de resoluções de problemas, com métodos considerados fracos, pois se houvesse falhas em alguma sequência dos métodos desenvolvidos não era possível encontrar a solução do problema.

A complexidade na solução de problemas foi se tornando desafiadora para os pesquisadores da época, pois solucionar problemas com base em métodos fracos não era mais eficiente [15], então, foi-se observado que seria necessário criar novos métodos por base no conhecimento já adquirido, para que a partir de então, pudesse construir sentenças em uma linguagem de representação, modelando-se assim o problema que se deseja solucionar. Desta forma, inicia-se a fase de sistemas baseados no conhecimento, o qual possuem em suas características uma base de conhecimento e um mecanismo de raciocínio, capazes de realizar inferências sobre esta base e obter conclusões a partir deste conhecimento [24].

Os sistemas especialistas surgem a partir da necessidade de processar dados não numéricos, eles utilizam uma combinação das duas áreas anteriores, a resolução de problemas e sistemas baseados em conhecimento. Um sistema especialista é baseado em conhecimento projetado para emular a especialização humana de algum domínio específico [25]. Eles são planejados para reproduzir o comportamento especialista humano resolvendo problemas do mundo real, mas com uma restrição no domínio do problema [26].

Desde 1980, surgiu uma nova IA mais moderna voltada ao comércio e fabricação de softwares mais sofisticados [15]. Apesar da IA moderna se expandir a partir de 1980, ela surgiu a partir de ideias anteriores da IA clássica.

A IA moderna considera basicamente:

- Aprendizado de máquinas;
- Recozimento simulado;
- Algoritmo Genético;
- Lógica *fuzzy*;
- Redes neurais artificiais;
- Busca tabu;
- Agentes inteligentes;
- Sistemas multiagente; IA distribuída;
- Computação evolutiva;
- Dentre outras que são criadas na atualidade.

Um dos primeiros artigos a abordar o uso da inteligência das máquinas foi proposto por Alan Turing, em 1950 [27]. Este artigo permanece atual em relação à sua ponderação dos argumentos e na possibilidade de ser possível criar uma máquina computacional

inteligente com respostas a esses argumentos. Turing é conhecido como um dos principais pesquisadores que contribuíram com a teoria da computabilidade, abordando a questão de uma máquina ser possível pensar ou não [28].

O teste de Turing analisa o desempenho que uma máquina pode ter em relação ao desempenho do ser humano em sua inteligência. O teste, também conhecido como o jogo da imitação, separa a máquina considerada inteligente do correspondente humano em duas salas diferentes. Entra então, uma terceira entidade, também humana, chamada de interlocutor que estará em uma terceira sala e irá interagir com ambos agentes através de um dispositivo textual, como um terminal. O interlocutor tentará distinguir quem é o homem e quem é a máquina, caso esta diferença não seja identificada, considera-se a máquina inteligente [28].

A inteligência artificial ao longo dos anos foi expandindo suas áreas de estudos, saindo do contexto de homem e máquina e explorando a natureza e outros comportamentos que fossem possíveis de se simular e contribuir ao encontro de soluções de problemas. O estudo de modelos biológicos e sociais da inteligência deu origem a novos algoritmos bioinspirados, a algoritmos que trabalham com redes neurais e lógicas racionais.

Algoritmos como recozimento simulado, que simula o esfriamento de metais líquidos para garantir uma baixa energia e formatos de estruturas altamente sólidas; algoritmos genéticos, que simula a evolução das espécies através da seleção, *crossover* e mutação; a lógica *fuzzy*, ou lógica difusa, que trabalha junto com a lógica multivalorada na qual se tem valores lógicos das variáveis próximos a qualquer número real entre zero ou um; redes neurais artificiais, que simulam o comportamento de um neurônio; e busca tabu, que guia um algoritmo de busca local na exploração contínua dentro de um espaço de busca, são algoritmos que deram início a uma nova abordagem de IA.

Com o passar do tempo, e com a evolução da IA, novos estudos foram surgindo. Começou-se o estudo de agentes inteligentes, dividindo-os em outras subáreas da IA, como sistemas multiagentes e IA distribuída. Pesquisas da evolução das espécies e dos algoritmos baseados no comportamento da natureza, base utilizada neste estudo proposto, criou os algoritmos principais destes estudos, como o PSO, ACO, GA, CS, bem como o algoritmo proposto nesta pesquisa, o *Locust Swarm Optimizer*.

2.3 Métodos de inteligência artificial bioinspirados

Nos próximos tópicos serão abordados conceitos e exemplos de algoritmos que tem por inspiração a natureza, assim como o comportamento das espécies que nela habita, são os chamados métodos de inteligência artificial bioinspirados.

2.3.1 Algoritmos evolutivos

Os algoritmos evolutivos, no inglês *Evolutionary Algorithm* (EA), correspondem a um conjunto de técnicas computacionais da computação bioinspirada [29] ou computação natural [30][31]. Estas técnicas computacionais abrangem fundamentos de conceitos biológicos inspirados na natureza, como as técnicas evolutivas que buscam inspiração em áreas da biologia em especial nas ideologias evolutivas e na genética.

Os EAs seguem fortemente os processos evolutivos que acontecem na natureza. Os principais componentes que possuem em um sistema evolutivo, segundo [32], são:

- Populações de indivíduos;
- Aptidão;
- Noção de mudanças dinâmicas;

- Conceitos de variabilidade e hereditariedade.

Nas populações de indivíduos uma ou mais destas populações estão concorrendo por recurso limitado. A habilidade que o indivíduo tem de sobreviver ou de se reproduzir é calculada na aptidão. Estes indivíduos podem nascer ou morrer, assim é necessário ter uma noção de mudanças dinâmicas que possam ocorrer na população. Os novos indivíduos têm características semelhantes à de seus pais, embora não sejam idênticos, assim vêm os conceitos de variabilidade e hereditariedade.

Por base nesta abordagem foram desenvolvidas técnicas de formas independentes inspiradas nos algoritmos evolutivos, como a programação evolucionária (*Evolutionary Programming* - EP), as estratégias evolutivas (*Evolutionary Strategies* - ES) e os algoritmos genéticos [33]. Os fundamentos básicos destas técnicas são basicamente as mesmas [34], inicialmente é gerada uma população de indivíduos (exemplo: um conjunto de possíveis soluções) e as influências do ambiente suscita um processo de seleção natural, ou seja, é um modo de garantir as melhores soluções que até então foram encontradas, isto causa um incremento no ajuste das soluções. Dando-se uma função objetivo a ser otimizada, gera-se um conjunto de soluções aleatoriamente (elementos que pertencem ao domínio desta função) e aplica-se esta função a fim de medir a qualidade das soluções requerentes, atribuindo nas soluções um valor para medir sua adequação, chama-se este valor de *fitness*.

Em seguida, selecionam-se algumas das melhores soluções até então encontradas por base no *fitness* e cria-se uma nova população de indivíduos através da aplicação de operadores de mutação e/ou recombinação. O operador de recombinação é aplicado a duas ou mais soluções requerentes, denominadas pais, e tem-se por resultado duas ou mais novas soluções denominadas descendentes ou filhos. O operador mutação é aplicado em

uma função candidata no intuito de gerar outra função, ao fim deste processo novas funções descendentes competem com as requerentes anteriores por base no *fitness*, a fim de assumir um lugar na nova população gerada. A iteração deste processo ocorre até que uma função requerente apresente uma solução seja satisfatória em termos de qualidade ou até que um número máximo de iterações (também denominados gerações) seja obtido.

Diversos componentes de um processo evolutivo dependem de uma variável aleatória, ou seja, são estocásticos. Isto faz com que a seleção favoreça indivíduos mais bem adaptados, com melhor *fitness*, mas há a possibilidade de serem escolhidos outros indivíduos. O processo de recombinação dos indivíduos ocorre de forma aleatória assim como o processo da mutação [34]. A Figura 2.1 representa de forma geral um pseudocódigo de um EA típico.

Pseudocódigo de um EA típico.

Input: Parâmetros típicos [35].
Output: População final de soluções
1 INICIALIZA população com soluções candidatas aleatórias
2 AVALIA cada candidata
3 **while** *critério de parada não é atingido* **do**
4 SELECIONA pais
5 RECOMBINA pares de pais
6 MUTA os descendentes resultantes
7 AVALIA novas candidatas
8 SELECIONA indivíduos para a nova geração
9 **end**

Figura 2.1- Pseudocódigo do EA (Adaptado de [35]).

2.3.2 Algoritmos genéticos

O estudo sobre os algoritmos genéticos ou *Genetic Algorithm* (GA), deu início na década de 40, quando pesquisadores buscaram na natureza inspiração para criar a inteligência artificial. Até meados de 50 as pesquisas relacionadas ao estudo tiveram um desenvolvimento maior nas áreas de pesquisa cognitiva e na interpretação de processos de raciocínio e aprendizado.

Na primeira metade da década de 60 Rechenberg [36] desenvolveu as estratégias evolucionárias, na qual foi considerado uma das primeiras tentativas a utilizar processos evolutivos para revolver problemas de otimização. Tais estratégias possuíam uma população com dois indivíduos possuindo cromossomos compostos de números reais em cada instante, de forma que um dos dois era filho do outro e sua geração se dava a partir da aplicação exclusiva do operador de mutação. O trabalho de Rechenberg pode ser considerado pioneiro por introduzir a computação evolucionária às aplicações práticas [37].

No final da década de 60, John Holland desenvolve os algoritmos genéticos, tornando-se assim o principal autor da descoberta. Através do estudo sobre a evolução das espécies, feito por Holland, o pesquisador conseguiu propor um modelo computacional que quando é implementado consegue encontrar boas soluções para problemas complexos e que eram insolúveis computacionalmente naquela época. Em 1975, Holland publica seu livro [38] sobre os estudos dos processos evolutivos, no qual aborda sobre os algoritmos genéticos [39] [37].

Os algoritmos genéticos se caracterizam por serem técnicas de busca que têm por inspiração a evolução natural das espécies e pertencem ao ramo dos algoritmos evolucionários [37]. Nos algoritmos genéticos faz-se a criação de uma população de

indivíduos e submete estes indivíduos aos operadores genéticos de seleção e de recombinação (*crossover*). Estes operadores fazem a avaliação de cada indivíduo, isto é, utilizam uma característica de qualidade de cada indivíduo como uma solução para o problema. Assim é gerado um processo de evolução natural dos indivíduos desta população criada, que em conseguinte irá gerar um indivíduo com características de uma boa solução para o problema abordado.

Na Figura 2.2 é apresentado um pseudocódigo do GA.

Pseudocódigo do *Genetic Algorithm*

Seja *P(x)* a população de cromossomos da geração *x*
x ←*0*
Inicializar *P(x)*
Avaliar *P(x)*
while o critério de parada não for satisfeito **do**
x ← *x* +1
Selecionar *P(x)* a partir de *P(x*-1*)*
Aplicar crossover sobre *P(x)*
Aplicar mutação sobre *P(x)*
Avaliar *P(x)*
end

Figura 2.2- Pseudocódigo do GA [37].

2.3.3 Otimização por enxame de partículas

A Otimização por Enxame de Partículas ou PSO (*Particle Swarm Optimization*), foi apresentada inicialmente na literatura [4] como uma alternativa inovadora para a otimização de funções não-lineares. O estudo do PSO teve por base o comportamento social de animais que vivem em colônias, como peixes e aves. Esta técnica de inteligência artificial busca imitar este comportamento.

No algoritmo PSO a solução para um problema de otimização está em algum espaço de busca n-dimensional. O algoritmo é inicializado com uma população inicial candidata a solução do problema denominada de partículas [40,41].

De forma semelhante à técnica dos algoritmos genéticos (AG), o PSO inicializa um enxame (*swarm*) com uma quantidade i de partículas, cada partícula possui uma dimensão d que representa as possíveis soluções do problema. Isto ocorre de modo em que todos os elementos do enxame estejam dentro de um intervalo pré-estabelecido $[x_{min}, x_{max}]$, assim como aquela solução com a melhor avaliação (avaliação global) que deverá orientar a busca no hiperespaço pela solução sub-ótima, ou seja, soluções que possuem valores aproximados ao ótimo da função. Os melhores valores individuais de cada partícula são armazenados, e assim, o mais bem avaliado irá representar a nova avaliação ótima se esse se sobrepuser àquela estabelecida na iteração anterior. Dessa maneira, cada partícula possui sua própria velocidade que será atualizada ao longo das iterações conforme os melhores valores individuais e o valor global do enxame para em seguida atualizar o valor de cada partícula, conforme é demonstrado em (2.1) e (2.2) [42]:

$$v_i^{t+1} = w * v_i^t + c_1 * r_1(p_i - x_i^t) + c_2 * r_2(g - x_i^t) \quad (2.1)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (2.2)$$

De modo que, v_i^t e v_i^{t+1} representam o vetor velocidade das partículas da posição v_i^{t+1} respectivamente na iteração t e $t+1$, w tem como definição o coeficiente da inércia, c_1 e c_2 são as constantes positivas, r_1 e r_2 são os valores arbitrários definidos no intervalo [0,1], p_i e g representam os vetores respectivamente da melhor solução relativa a posição i e da melhor solução global, e por fim, x_i^t e x_i^{t+1} representam o vetor partícula na posição i do enxame respectivamente nas iterações t e $t+1$.

Para encontrar a nova velocidade, devem-se saber os limites da velocidade. Isto é, a velocidade máxima v_{max} deverá ser igual a x_{max} e a velocidade mínima v_{min} será igual a x_{min}.

O grau de independência e dependência das partículas é definido pelas constantes positivas c_1 e c_2, respectivamente, isto é, c_1 é o coeficiente cognitivo e corresponde ao grau de liberdade que uma partícula possui ao percorrer o espaço de busca a procura do seu próprio valor ótimo e c_2 é o coeficiente social, que representa o grau de cooperação entre as partículas para chegar até o valor ótimo global. Normalmente estes valores se relacionam tal como definido em (2.3).

$$c_2 = 4 - c_1 \tag{2.3}$$

Por fim, o coeficiente da inércia w indica o quanto a velocidade v_i^t irá interferir na nova velocidade v_i^{t+1}. Na literatura, existem diversos tipos de estratégias em relação à escolha do coeficiente de inércia, uma delas é o coeficiente com decaimento linear [43]. Para esta estratégia são definidos os valores de coeficiente de inércia inicial ($w_{inicial}$) e ao longo das iterações t o coeficiente será atualizado de forma decrescente conforme a quantidade t_{max} de iterações até atingir o valor final (w_{final}) definido para última iteração, conforme demonstrado em (2.4).

$$w = w_{final} + \frac{(w_{inicial} - w_{final})(t_{max} - t)}{t_{max}} \tag{2.4}$$

Com base na descrição desta seção, a Figura 2.3 representa a descrição do algoritmo PSO discorrido nesta seção.

Pseudocódigo do PSO.
1. Determine o número de partícula p da população Pop.
2. Inicialize aleatoriamente a posição inicial (x_i^t) de cada partícula p de Pop.
3. Atribua uma velocidade inicial (v_i^t) igual para todas as partículas.
4. Para cada partícula p em Pop faça:
(a) Calcule sua aptidão fitness = $f(p)$.
(b) Calcule e melhor posição da partícula p até o momento (p_{best}).
5. Descubra a partícula com a melhor aptidão de toda a população (g_{best}).
6. Para cada partícula p em Pop faça:
(a) Atualize a velocidade da partícula pela fórmula da equação (2.1)
(b) Atualize a posição da partícula pela fórmula da equação (2.2)
7. Se condição de término não for alcançada, retorne ao passo 4.

Figura 2.3- Pseudocódigo do PSO [42].

2.3.4 Otimização por colônias de formigas

A otimização por colônias de formigas, em inglês *Ant Colony Optimization* (ACO), tem por inspiração o comportamento de formigas artificiais, considerando a experiência acumulada pela colônia. Esta experiência pode ser presentada por trilhas de feromônio e informações heurísticas que são específicas ao problema a ser resolvido [44].

O algoritmo ACO foi proposto pela primeira vez por Marco Dorigo, em sua tese de doutorado na Itália em 1992. O algoritmo tem por abordagem o uso da inteligência coletiva e foi proposto primeiramente para a resolução do problema do caixeiro viajante (*Travelling Salesman Problem* - TSP). No TSP, um único caixeiro viajante deve visitar um conjunto de *n* cidades, passando por ela apenas uma única vez até voltar à cidade de origem novamente. O ACO procura imitar o comportamento de formigas reais saindo de seus ninhos até uma fonte de alimento, conforme ilustra a Figura 2.4.

Nesta Figura 2.4 as formigas conseguem encontrar o melhor caminho até a fonte de alimento, através de substâncias químicas chamadas de feromônios. As formigas ao longo do caminho percorrido liberam o feromônio em sua trilha, desta forma quanto menor o

caminho percorrido por um grupo de formigas maior será a quantidade de feromônio concentrado na trilha, fazendo com que as formigas sigam este caminho. E, por conseguinte, quanto maior o caminho até a fonte de alimentos menor será a quantidade de feromônios deixado na trilha, fazendo com que as formigas deixem de escolher este caminho, e buscar um caminho com maior concentração de feromônios. Estes caminhos são escolhidos de forma aleatória e os piores (com menor quantidade de feromônios) são abandonados pelas formigas, devido ao feromônio ali depositado evaporar com o tempo [45].

O ACO possui versões de problemas de forma discreta e de forma contínua. Para este trabalho foi implementada a versão contínua do algoritmo, por base na literatura de Socha e Dorigo [46].

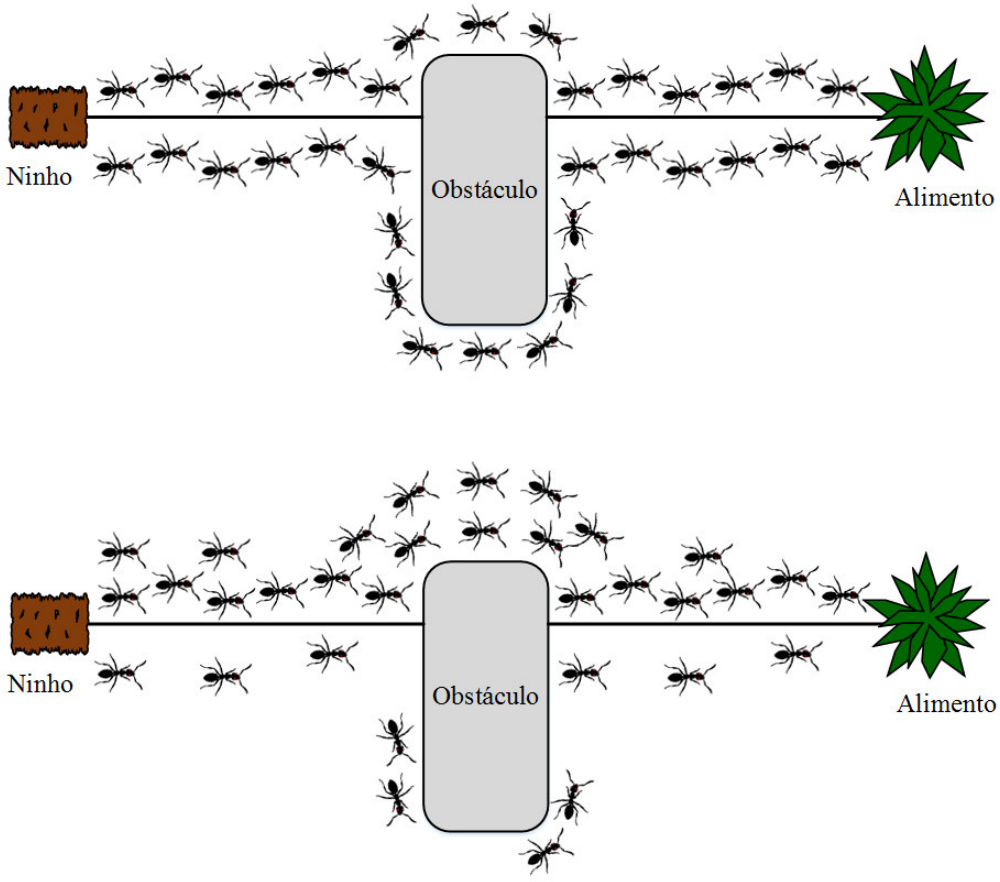

Figura 2.4- Representação do caminho feito pelas formigas do ninho até a fonte de alimento.

O algoritmo ACO deve-se iniciar com uma população de formigas de maneira aleatória, e então, armazenar essa população em uma matriz $pop_{n\ X\ dim}$. De modo que n representa o tamanho da população e dim a dimensão do problema a ser resolvido. Na sequência, em todo vetor da população pop_i $(i = 1, \dots, n)$ deve-se associar um peso w_i conforme a função Gaussiana com média $u = 1$ e desvio padrão $\sigma = q \times n$, onde o valor de q representa o parâmetro do algoritmo. Se o valor q for pequeno, a probabilidade de escolha das soluções melhores é maior, ao contrário, se o valor de q for maior, a escolha tenderá a ser uniforme [46]. Assim, a probabilidade de seleção do elemento pop_i é representada em (2.5):

$$p_i = \frac{w_i}{\sum_{r=1}^{n} w_r} \tag{2.5}$$

Na seguinte etapa deve-se encontrar o desvio padrão σ_i equivalente a cada elemento do vetor de soluções de dimensão dim. O desvio padrão σ_i é empregado no algoritmo em (2.6):

$$\sigma_i = \frac{\zeta}{n-1} \sum_{\substack{l=1 \\ l \neq r}}^{n} |pop_l - pop_r| \tag{2.6}$$

Onde, ζ representa a taxa de evaporação do feromônio.

O próximo passo é a criação da nova população, representada pelo vetor $newpop_{sample\ X\ dim}$. Onde, $newpop$ possui $sample$ linhas, representando o tamanho da amostra; e dim colunas. É através da escolha um elemento da população, que se torna possível calcular os valores da nova população. Esta escolha é feita através do método de roleta russa e o elemento $newpop_{ij}$ é dado em (2.7).

$$newpop_{ij} = pop_{tj} + \sigma_{tj} \times rand() \quad (2.7)$$

Onde t representa um índice que é gerado pela roleta russa.

No final, calcula-se o valor do *fitness* da nova população, e compara-se com o valor da população antiga. Se o *fitness* desta nova população for melhor do que o *fitness* da antiga, então, $pop = newpop$. A Figura 2.5 resume o procedimento do algoritmo ACO.

Pseudocódigo do ACO	
1	Parâmetros de entrada: ζ, q e tamanho amostra t.
2	Gerar população inicial pop aleatoriamente
3	Cálculo do peso w
4	Cálculo da probabilidade de seleção p
5	Cálculo do desvio padrão σ
6	Formulação da nova população $newpop$
7	Concatenar $newpop$ com pop
8	Selecionar os melhores fitness
9	Se o critério de parada for atingido ir para o passo seguinte, caso contrário voltar para o passo 5.
10	Melhor indivíduo da população pop_{best}

Figura 2.5- Pseudocódigo do ACO [46].

2.3.5 Busca do pássaro cuco

O algoritmo do *Cuckoo Search* (CS), ou na tradução busca do pássaro cuco, baseia-se no comportamento do parasitismo da ninhada de determinadas espécies de cucos. Na natureza existem três tipos básicos de parasitismo das ninhadas de aves, são elas, o intraespecífico parasitismo da ninhada, a criação de cooperativismo, e a aquisição do ninho. Algumas aves da região podem entrar em conflitos com os cucos intrusos, buscando proteger seu ninho. Também, pode ocorrer a descoberta que os ovos não pertencem à ave do ninho e a mesma poderá jogar fora estes ovos, ou simplesmente abandonar o ninho para construir outro [3].

Na natureza existem espécies de cucos, como a Tapera, que ao longo dos anos evoluiu sua estratégia de depósito de ovos em ninhadas de outras aves, de tal forma, que é

capaz de produzir ovos com a mesma cor e padrão das espécies hospedeiras escolhidas para desovar seus ovos [47]. Devido ao instinto dos animais, uma vez que os pássaros hospedeiros choquem os ovos dos cucos, os hospedeiros alimentam e tratam os filhotes dos cucos como se fossem seus próprios filhotes aumentando assim sua probabilidade de sobrevivência.

Por base nesse comportamento dos cucos e da rejeição das aves dos ovos do parasita foi criado o algoritmo do *Cuckoo Search*. O estudo deste algoritmo está associado também ao comportamento de voos de muitos animais e insetos na natureza, que possuem características de voos de Lévy. Os voos de Lévy caracterizam-se pela construção de trajetórias curtas e longas predominantemente curtas [48]. A Figura 2.6 mostra os passos de um pseudocódigo do CS na otimização de problemas.

Pseudocódigo do *Cuckoo Search*.

```
1  begin
2      Função objetiva f(x), x = (x_1, ..., x_d)^T;
3      Iniciar uma população de n ninhos de acolhimento x = (i = 1,2, ..., n);
4          while (t < Geração Máxima) ou (critério de parada);
5              Obter um cuco (digamos i) aleatoriamente
6                      e gerar uma nova solução de voos de Lévy;
7              Avaliar a sua qualidade / fitness; F_i
8              Escolha um ninho entre n (digamos j) aleatoriamente;
9              if (F_i > F_j),
10             Substitua j pela nova solução;
11             end
12             Abandonar uma facção (P_a) de ninhos piores
13                     [E construir novos em novos locais através de voos de Lévy];
14             Manter as melhores soluções (ou ninhos com soluções de qualidade);
15             Classificar as soluções e encontrar a atual melhor;
16         end while
17         Publicar resultados do processo e visualização;
18 end
```

Figura 2.6- Pseudocódigo do método *Cuckoo Search* [49].

2.3.6 Sistemas imunológicos artificiais

Os sistemas imunológicos (*Immune Systems* - IS) têm por objetivo inspecionar e analisar células que não se comportam de forma correta no organismo humano, como exemplo os tumores, ou procurar elementos externos que causam doenças ao organismo, como os vírus e as bactérias, assim, sua função é corrigir anomalias e defender o organismo de possíveis ataques [50].

Antígenos são todos os elementos em que o sistema imunológico reconhece. Antígenos próprios são células do organismo humano que não são prejudiciais ao organismo. Antígenos não-próprios são as células que são prejudiciais ao organismo. O sistema imunológico consegue distinguir quais são os antígenos próprios e não-próprios do organismo humano [50].

Recentemente existem inúmeras pesquisas na área médica tentando compreender o funcionamento dos sistemas imunológicos, a fim de combater e prevenir doenças infectuosas, por exemplo, a AIDS, doenças autoimunes e outros problemas relacionados à imunidade do corpo humano. Além da medicina outras áreas, como engenharia e computação, pesquisam sobre o tema. Cientistas da computação buscam simular mecanismos imunológicos com o intuito de criar sistemas artificiais para busca de soluções de problemas complexos da área [51].

A capacidade de processamento de informações fornecida pelos sistemas imunológicos (SI) vem trazendo forte interesse nas áreas de computação e engenharia. Algumas características do SI servem de destaques para áreas citadas. A unicidade, o reconhecimento de padrões internos e externos ao sistema, a detecção de anomalia, a detecção imperfeita (tolerância a ruídos), a diversidade, a aprendizagem por reforço e a

memória, estas características emuladas em SI vêm solucionando problemas computacionais complexos em diversas pesquisas da área [52-54].

Na literatura há técnicas que buscam simular o princípio de seleção clonal nas quais se aplicam em problemas de aprendizagem de máquina, reconhecimento de padrões e na otimização [55]. O CLONALG (*CLOnal selection ALGorithm*, ou algoritmo de seleção clonal), proposto por [56,57], imita o princípio de seleção clonal aplicado tanto na otimização como em problemas de aprendizado de máquinas. Um modelo de pseudocódigo do CLONALG é descrito na Figura 2.7 baseado na referência [57].

Pseudocódigo do CLONALG quando aplicado a problemas de otimização.

```
input: β, ρ, tamanhoPopulacao, nAleatorio, nSelecao
output: anticorpos
1 begin
2       anticorpos ← inicializaPopulacao(tamanhoPopulacao);
3       afinidades ← avalia(anticorpos);
4       while critério de parada não é atingido do
5       afinidadesNormalizadas ← normaliza(afinidades);
6       selecionados ← seleciona(anticorpos, afinidadesNormalizadas, nSelecao);
7       clones ← clona(selecionados, afinidadesNormalizadas, β);
8       hipermuta(clones, afinidadesNormalizadas, ρ);
9       afinidadesClones ← avalia(clones);
10      substituiPopulacao(anticorpos, afinidades, clones, afinidadesClones,
        nSelecao, nAleatorio);
11      end
12 end
```

Figura 2.7- Pseudocódigo do CLONALG quando aplicado a problemas de otimização [56].

O pseudocódigo, presente na Figura 2.7, tem por parâmetros de entradas passadas pelo usuário um valor β no qual determina o número de clones que são gerados por cada anticorpo, um fator de ajuste ρ que ajusta a intensidade da hipermutação aplicadas sobre os clones, o tamanho *tamanhoPopulacao* de soluções requerentes da população anticorpos gerada, o número *nAleatorio* de anticorpos que possuem a menor afinidade e que devem

ser substituídos por anticorpos novos aleatoriamente gerados, por fim, a quantidade *nSelecao* de anticorpos a serem escolhidos para o processo de clonagem. Em seguida o algoritmo inicializa até que seu critério de parada é atingido. No fim serão apresentados à quantidade de anticorpos otimizados [56,57].

A seguir, por base no estudo feito neste capítulo 2 no qual foram abordados alguns dos principais algoritmos bioinspirados estudados na literatura, será discutido no capítulo 3 a proposta de um novo método de inteligência baseado no comportamento de enxames dos gafanhotos-do-deserto e a criação deste método.

3 Método proposto de inteligência artificial do gafanhoto-do-deserto

3.1 Introdução

Neste capítulo, um modelo matemático que tem por base o comportamento de um enxame de gafanhotos-do-deserto, sua organização para a formação de nuvens de ataques e um algoritmo resultante deste comportamento, o *Locust Swarm Optimizer* (LSO), será proposto.

Na literatura, Ribeiro [58,59] propõe ideias de criações de algoritmos que têm por base o comportamento do enxame dos gafanhotos-do-deserto. Anstey e Rogers [60,61] descrevem que o principal fator para a formação do enxame (nuvem de ataques) desse inseto se dá devido à elevação da taxa de uma monoamina neurotransmissora chamada serotonina.

O modelo proposto descreve estes passos da estrutura da formação da nuvem de ataques e foi modelado matematicamente para ser testado posteriormente.

3.2 Motivação

A motivação deste estudo vem por base nas propostas feitas por [58,59], que propõem uma arquitetura computacional voltada ao comportamento do gafanhoto-do-deserto a fim de resolver problemas complexos, tanto na rede computacional, como nas nuvens. Por base nesta literatura e para melhor entendimento, foram-se abstraindo as informações que

melhor se adequam a otimização de sistemas e feito um levantamento bibliográfico do comportamento deste inseto, conforme a seguir.

Gafanhotos são pragas que devastam enormes plantações, algumas espécies são migratórias, outras passam solitárias durante seu período de vida. A formação da nuvem de ataque é o principal interesse neste estudo, pois há um comportamento que pode ser aplicado na área de otimização, este comportamento é a principal inspiração deste estudo.

Os gafanhotos são conhecidos também como acrídeos, ticuras, acridianos ou tucuras, insetos pertencentes à subordem *Caelifera* da ordem *Orthoptera*. Tem como características o fémur das pernas posteriores muito grandes e fortes, o que lhe fornecem o deslocamento através de saltos. Algumas espécies podem formar grandes enxames podendo acabar com grandes plantações, em pouco tempo [62].

As características da subordem dos gafanhotos que diferem da subordem *Ensifera*, são as antenas pequenas, tamanho variável entre 1,5 centímetros e aproximadamente 10 centímetros, tarsos com três segmentos ou menos, ovopositor curto e tímpanos presentes dos lados do primeiro segmento abdominal [63].

Os gafanhotos são fitófagos, ou seja, se alimentam de vegetais, sendo considerados nocivos à vegetação. Dessas espécies podem existir indivíduos solitários que nunca formam aglomerados e indivíduos que se reúnem formando grandes bandos, conhecidos como nuvens de gafanhotos, que migram para regiões distantes em busca de plantações que encontram em sua frente [63].

Na América do Sul, a espécie responsável por este tipo de ataque é a *Schistocerca Americana* da subfamília *Cyrtacanthacridinae*. É uma espécie migratória com cerca de 45

a 66 mm de comprimento, de coloração pardo-avermelhada e que se espalha causando grandes danos à agricultura, principalmente no sul do Brasil [63].

Os gafanhotos são conhecidos também na histologia bíblica, no qual narra que no antigo Egito uma nuvem de gafanhoto devastou todo o alimento da população egípcia, acabando com toda a plantação presente na região, causando fome e mortes em toda a região. Tal fato ocorreu por naquela época não existir equipamentos modernos e usos de pesticidas que combatesse a praga. Ao longo doa anos, a ciência evoluiu e hoje existem trabalhos voltados ao combate da praga, o que pode evitar grandes devastações como ocorreu no antigo Egito.

3.3 O gafanhoto-do-deserto

O gafanhoto-do-deserto possui uma característica de formação de nuvens de ataques durante seu ciclo de vida, para a formação de seu enxame de ataque esta espécie passa por duas fases de transformação [60,61]. Este comportamento será abordado nos tópicos a seguir.

3.3.1 Características do gafanhoto-do-deserto

Os gafanhotos são espécies que podem ser encontrados em diversas regiões do mundo e pertencem à família de insetos *Acrididae* (*Orthoptera*: *Caelifera*). De acordo com o seu comportamento eles podem ser divididos em duas categorias: as espécies sedentárias ou solitárias e pouco nocivas; e as migratórias, que apresentam hábito gregário formando as chamadas “nuvens de gafanhotos”. Os gafanhotos do deserto, da espécie *Shistocerca Gregária*, se destacam dos demais gafanhotos por viverem em uma fase inicialmente solitária e devido a um estímulo (encontro do alimento ou outro gafanhoto de sua espécie) ele passa para a fase de agregação [64].

Pesquisadores observaram [60, 61, 64], que na fase adulta o gafanhoto-do-deserto possui um período de sua vida solitário onde possui uma cor esverdeada (Figura 3.8), mas ao encontrar abundância de alimento, uma monoamina neurotransmissora sintetizada chamada de serotonina, presente nele se eleva, fazendo com que o inseto mude sua fase inicial solitária para uma fase de agregação (em busca de outros insetos de sua espécie) e mudando sua coloração para amarelada (Figura 3.9) [65,60].

Figura 3.8- Gafanhoto-do-deserto na fase solitária.

Figura 3.9- Gafanhoto-do-deserto na fase gregária.

No desenvolvimento do estudo foi descoberto que existem dois aspectos da concentração de gafanhotos que provocam sua transformação e a serotonina estaria por trás de ambos os aspectos. Primeiro, foi observado que nas disputas entre os gafanhotos eles se chocam contra os pêlos tácteis em suas patas traseiras, que provocam a produção de serotonina, e o segundo aspecto observado é que a percepção de outros gafanhotos através da visão e do cheiro também produz serotonina. Estes dois percursos são muito diferentes, mas convergem no gânglio torácico que recebe sinais a partir do cérebro e das pernas [65].

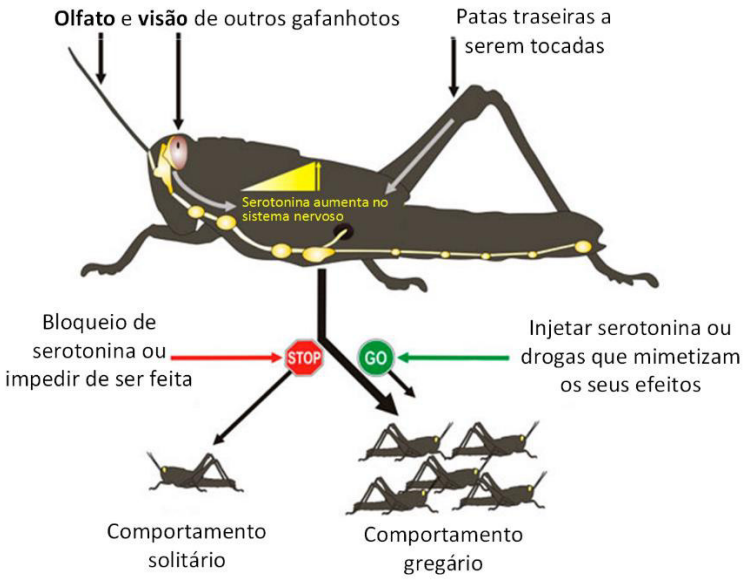

Figura 3.10- Forçamento de mudanças de fases de gafanhotos em fase solitária para fase de gregária (Adaptado de [66]).

A Figura 3.10 representa os experimentos em laboratórios com os gafanhotos, onde houve aplicações de serotonina diretamente no gânglio torácico dos insetos em estudo ou uso de outros produtos químicos no ambiente em que os gafanhotos se encontravam que provocou o aumento da serotonina nos gafanhotos, o suficiente para que eles mudassem para o estado de agregação (ataque) [65,67].

Como já visto, esta fase gregária é responsável pela formação da nuvem de ataque e a consequente devastação de lavouras e plantações de agricultores pelo mundo, causando prejuízos a donos de fazendas e a economia do país.

3.3.2 Formação do enxame dos gafanhotos-do-deserto

O gafanhoto-do-deserto em sua procura por alimento passa por fases e transformações ao longo de sua busca. Cientistas pesquisaram o que motivou a busca desse alimento e como se procede à transformação do mesmo, quais fatores levam a essa mudança e como a serotonina atua no organismo desses insetos [60,61,65,67].

Com base nos estudos feitos por Anstey [60,61,65,67], a respeito de como o gafanhoto se transforma durante o ataque. A seguir é apresentado um breve esquema comportamental da formação das nuvens de ataques.

Figura 3.11- Saída à procura de alimentos.

O gafanhoto-do-deserto vive a priori uma fase solitária, seu nível de serotonina não é tão elevado e não pensa em estar em grupo. Ao longo do território em que vive este gafanhoto pode sair em busca de alguma fonte de alimento (conforme Figura 3.11), isso se dá através de voos na região usando a visão e o olfato para localizar o alimento.

Figura 3.12- Mudança de fase solitária para fase de gregária.

Quando o alimento é encontrado, o gafanhoto sai ao redor da região em que vive à procura de outros gafanhotos. Devido aos pulos, o atrito com as patas e o uso da visão e do olfato em busca de outros gafanhotos, a taxa de serotonina em seu corpo se eleva. Isso provoca uma transformação do gafanhoto de sua fase inicial solitária quando o inseto tem uma coloração corporal esverdeada, para a fase de agregação, ou fase de formação de

nuvens, na qual o gafanhoto-do-deserto mudará sua coloração corporal para amarelada. Como visto na Figura 3.12.

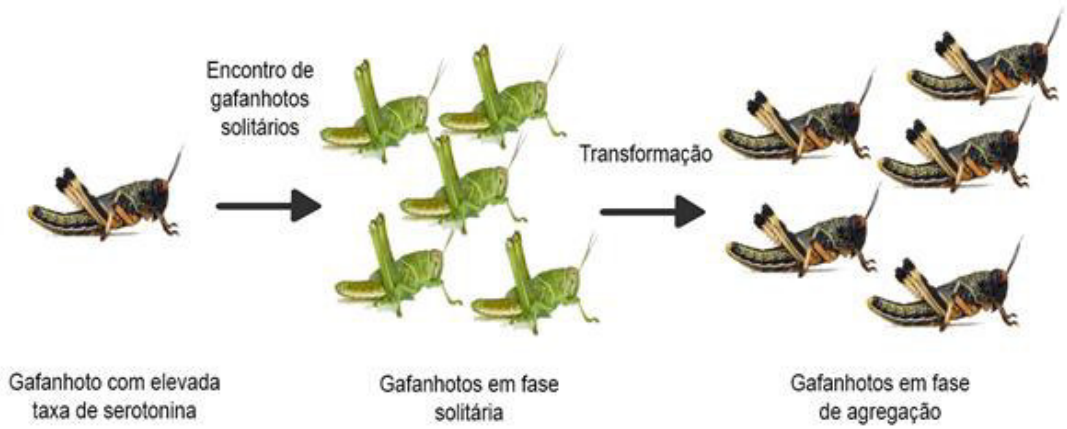

Figura 3.13- Estímulos de mudança em outros gafanhotos-do-deserto.

A Figura 3.13 apresenta como ocorre a transformação de outros gafanhotos que estavam inicialmente na fase solitária. O inseto que encontrou o alimento através dos atritos das patas estimula outros gafanhotos a passarem para a fase de agregação. Através da visão e do olfato esses outros gafanhotos aumentam sua taxa de serotonina e mudam de fase, segundo Anstey [60,61,65,67] o tempo dessa transformação é de, no máximo, duas horas.

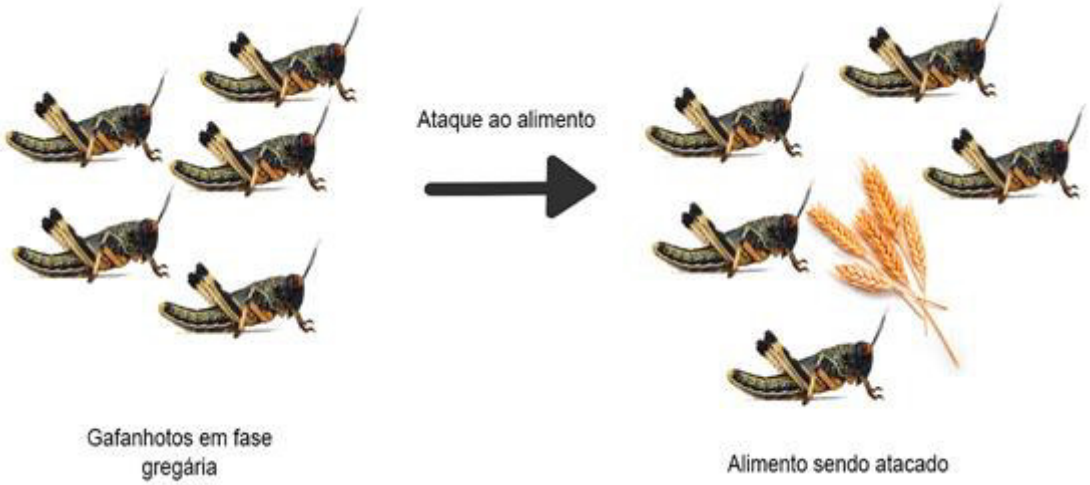

Figura 3.14- Ataque ao alimento.

Quando há um número considerável de gafanhotos em fase gregária para realizar o ataque ao alimento, eles o iniciam, conforme a Figura 3.14. A plantação é completamente ou parcialmente devastada, o que prejudica os agricultores e pecuários da região afetada.

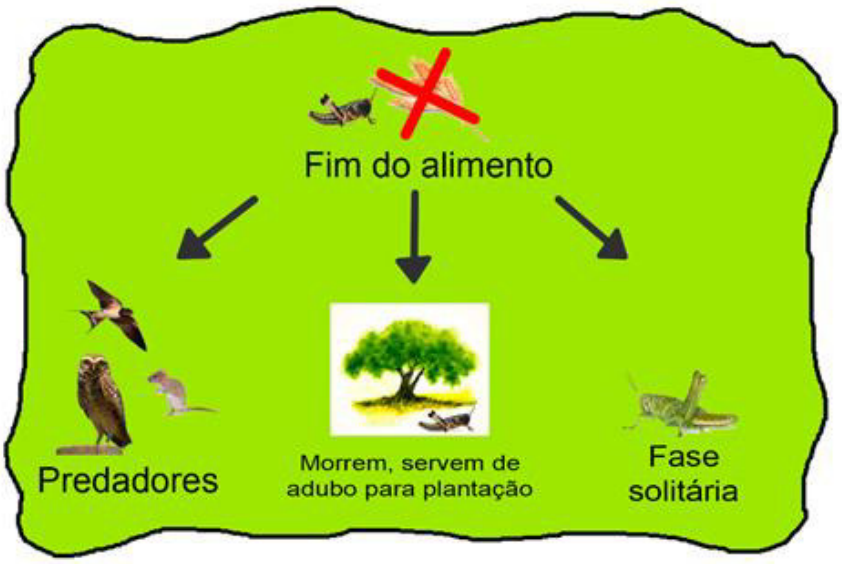

Figura 3.15- Direção dos gafanhotos após o alimento acabar.

Ao consumirem o alimento, os gafanhotos se dispersam ao redor da região, alguns morrem, vítimas de outros predadores, no qual servem como adubo para a terra e plantações ao redor. Os gafanhotos sobreviventes podem voltar à fase solitária na região [68], conforme a Figura 3. 15.

3.3.3 Algoritmo computacional do gafanhoto-do-deserto

No algoritmo LSO, os gafanhotos-do-deserto são representados por pontos, que voam ou se deslocam em um espaço de busca $\Re^{dim}$, onde dim é a dimensão do espaço. O modelo matemático é baseado no comportamento dos gafanhotos-do-deserto na natureza.

- Parâmetros de entrada

Semelhantemente as outras técnicas de IA, o algoritmo LSO possui alguns parâmetros de entrada. O nível de serotonina nos gafanhotos foi representado por uma

matriz S, cuja dimensão é $N_p \times Iter_{max}$, onde N_p é o tamanho da população e $Iter_{max}$ é o número máximo de iterações exigido no algoritmo. Inicialmente, todos os gafanhotos estão com nível de serotonina nulo. O tamanho da nuvem é representado por um escalar G. A princípio, o parâmetro G é igual a dois, variando no decorrer das iterações.

- Fase Solitária

A fase solitária do gafanhoto caracteriza-se por viver sozinho, desta maneira, o algoritmo inicializa com os gafanhotos na fase solitária. A posição inicial dos gafanhotos é dado em (3.1). Esta inicialização é feita através de um número aleatório, no qual é uma característica comum em todos os algoritmos de IA moderna.

$$x_{t=0} = (ub - lb) \times rand + lb \tag{3.1}$$

Onde x_0 denota a posição inicial dos gafanhotos; ub e lb são os limites superiores e inferiores das variáveis de decisão, respectivamente; t é a iteração corrente; e $rand$ é um número uniforme aleatório.

- Atualizar o nível de serotonina

A serotonina é a substância responsável pela formação das nuvens de gafanhotos-do-deserto. A principal diferença entre os gafanhotos na fase solitária e na fase gregária é a quantidade de serotonina [60]. O modelo matemático utilizado para atualização do nível de serotonina é apresentado em (3.2). Onde p é o nível de serotonina normalizado (3.3); t é a iteração corrente; e é a base do logaritmo natural e $1 \leq i \leq N_p$. Esta formulação da serotonina foi feita por base em testes de valores da serotonina estudada por Anstey [60].

$$S_t(i) = [1 - p(i)] \times G_t(i) \times e^{p(i)} + S_{t-1}(i) \tag{3.2}$$

$$p(i) = \frac{S_t(i)}{\max S_t} \tag{3.3}$$

- Atualizar a posição dos gafanhotos

A atualização da posição dos gafanhotos depende do nível de serotonina e da posição do gafanhoto mais próxima do objetivo (alimento). A atualização da posição dos gafanhotos na iteração t é dada em (3.4). Caso não encontre alimento, o gafanhoto retorna para fase solitária. Onde $1 \leq i \leq N_p$, $1 \leq j \leq dim$ e dim é a dimensão do problema.

$$x_t(i,j) = p(i) \times x_{t-1}(i,j) + rand \times [x_{t-1_{best}}(1,j) - x_{t-1}(i,j)] \quad (3.4)$$

- Atualizar o tamanho da nuvem.

As nuvens de gafanhotos podem reunir milhões destes insetos e são formadas por gafanhotos na fase gregária. O tamanho da nuvem obedece à equação (3.5) e depende da mudança no nível de serotonina do gafanhoto e da iteração corrente. Caso o gafanhoto não altere ou diminua seu nível de serotonina significa que ele retornou para a fase solitária.

$$G_t(i) = 2 \times t^{1.2} \quad (3.5)$$

A melhor posição e o melhor *fitness* serão armazenadas e guardadas para a iteração solicitada. Finalizando assim o algoritmo e localizando o melhor resultado.

A Figura 3.16 apresenta a proposta de um pseudocódigo do LSO através do modelo matemático criado, e voltado à otimização de sistemas. Este pseudocódigo apresenta etapas do comportamento do gafanhoto-do-deserto para otimizar problemas computacionais.

Inicialmente em um espaço de busca são colocados gafanhotos de forma aleatória, de forma que sua serotonina inicial é nula, calculam-se seus *fitness* e suas melhores posições no ambiente para que em seguida possa-se iniciar um *loop* do comportamento dos insetos.

No *loop* é atualizado o valor da serotonina, escolhendo um gafanhoto de forma aleatória para que ocorra esta atualização. Se o gafanhoto escolhido for uma possível solução ótima para problema a se resolver, então será atualizada sua taxa de serotonina e o

gafanhoto passará para a fase gregária (fase de formação de enxames), caso contrário, o gafanhoto permanecerá com sua taxa anterior (na sua fase solitária).

Em seguida, serão atualizadas as posições dos gafanhotos, de forma que separe os gafanhotos solitários dos gregários. O passo seguinte é atualizar a matriz G, onde serão armazenados os gafanhotos gregários (as melhores soluções encontradas). Esta atualização dependerá da taxa de serotonina dos gafanhotos, uma vez que a principal diferença entre o gafanhoto solitário para o gafanhoto gregário é sua taxa de serotonina. Por fim calcula-se o *fitness*, armazena-se o melhor *fitness* e sua respectiva iteração, e então finaliza o algoritmo LSO.

Pseudocódigo: Algoritmo *LSO*

Parâmetros de Entrada:

- Objectivefunction: $F(x)$.
- N_p: número inicial de gafanhotos (tamanho da população)
- $Iter_{max}$: número máximo de iterações;
- *lb*: limite inferior das variáveis de decisão;
- *ub*: limite superior das variáveis de decisão;
- *dim*: dimensão do problema;
- Tamanho inicial da nuvem: $G = 2 \times ones(N_p, 1)$;
- Inicializar p aleatoriamente: $p = rand(N_p, 1)$;
- Inicializar a matriz $S = zeros(N_p, Iter_{max})$;

1.Inicializar a posição dos gafanhotos. Gafanhotos na fase solitária. Eq. (3.1).

2.Cálculo do fitness para cada gafanhoto na fase solitária. Salvar melhor fitness e melhor posição.

while (t <$Iter_{max}$) **do**

1. Atualizar o nível de serotonina dos gafanhotos e escolher um gafanhoto *k* aleatoriamente.
   for *i* = 1 to N_p **do**
   if fitness(*i*)<fitness(*k*) **then**
   Atualizar o nível de serotonina. Eq. (3.2)
   else
   Serotonina permanece a mesma.
   end if
   end for
2. Atualizar a posição dos gafanhotos.
   for *i* = 1 to N_p **do**
   for *j* = 1 to *dim* **do**
   if $S_{(t)} - S_{(t-1)} > 0$ **then**
   Atualizar a posição *x(i,j)* dos gafanhotos. Fase gregária. Eq. (3.4)
   else
   Atualizar a posição *x(i,j)* dos gafanhotos. Fase solitária. Eq. (3.1).
   end if
   end for
   end for
3. Atualizar o tamanho da nuvem G. Atualizar a serotonina normalizada p.
   Atualizar p. Eq. (3.3).
   for *i* = 1 to N_p **do**
   if $S_{(t)} - S_{(t-1)} > 0$ **then**
   Aumentar o tamanho da nuvem de gafanhotos. Eq. (3.5)
   else
   Acabar com a nuvem de gafanhotos. G igual a dois.
   end if
   end for
4. Cálculo do fitness.
5. Armazenar o melhor fitness e a correspondente posição x_{best}.
6. Incrementar o contador: $t = t + 1$.

end while

9.return: Solução do problema x_{best} e valor da função objetivo $F(x_{best})$.

Figura 3.16- Proposta de um pseudocódigo do LSO.

4 Aplicações do método do gafanhoto-do-deserto

4.1 Introdução

Neste capítulo os testes e resultados com o método criado, o otimizador baseado em enxames de gafanhoto-do-deserto ou LSO (*Locust Swarm Optimizer*), serão abordados. O LSO será avaliado em 23 funções teste (funções *benchmark*), e também buscará resolver um problema de despacho econômico. Os resultados serão comparados com outros métodos clássicos de inteligência artificial existentes na literatura.

4.2 Funções teste a ser otimizadas

O algoritmo *Locust Swarm Optimizer* será avaliado em 23 funções *benchmarks*, funções as quais são muito utilizadas para testes em algoritmos que trabalham com programação evolutivas ou com exames, o artigo do Yao e Liu [69] utiliza funções *benchmarks* para testar um novo algoritmo evolutivo e avaliar seu desempenho, taxas de convergências e outros parâmetros comparando-os a outras técnicas da literatura, estas funções clássicas são muito utilizadas por pesquisadores [70-74]. Apesar da simplicidade, foram escolhidas estas funções teste para ser capaz de comparar os resultados com outros métodos de IA já existentes na literatura. Estas funções *benchmarks* estão listadas nas Tabelas 4.2-4.4, onde *Dim* indica dimensão da função, *Range* é o limite do espaço de busca da função, e f_{min} é o valor ótimo da busca. As Figuras 4.17- 4.19 ilustram as versões 2D das funções de referência usadas.

4.2.1 Funções teste *unimodais*

Tabela 4.2- Funções *benchmarks unimodais*.

Função	*Dim*	Range	f_{min}
$F_1(x) = \sum_{i=1}^{n} x_i^2$	10	[-100,100]	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10,10]	0
$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	30	[-100,100]	0
$F_4(x) = \max_i \{ \lvert x_i \rvert, 1 \leq i \leq n \}$	30	[-100,100]	0
$F_5(x) = \sum_{i=1}^{n} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	30	[-30,30]	0
$F_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[-100,100]	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1]$	30	[-1.28, 1.28]	0

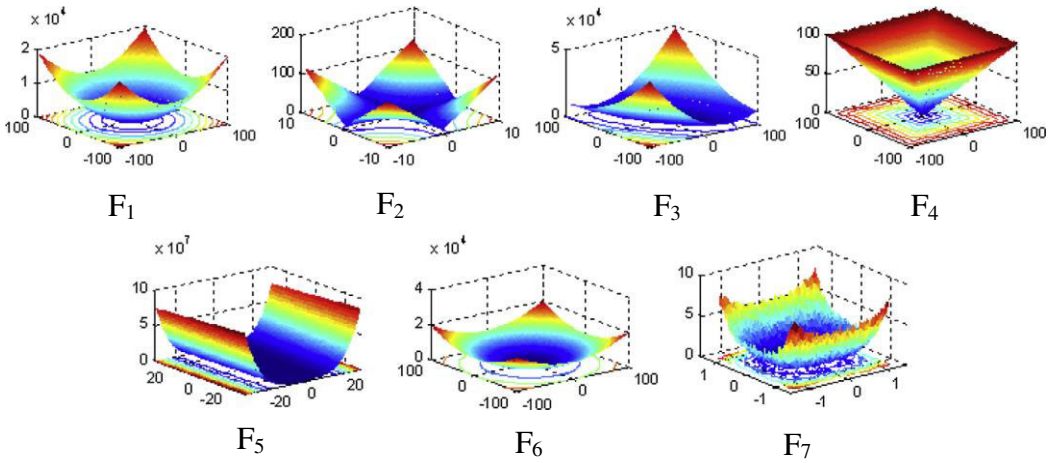

Figura 4.17- Versões 2D das funções *benchmarks unimodais*.

4.2.2 Funções teste *multimodais*

Tabela 4.3- Funções *benchmarks multimodais*.

Função	*Dim*	Range	f_{min}
$F_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	[-500, 500]	-418.9829 x *5*
$F_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	30	[-5.12, 5.12]	0
$F_{10}(x) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	[-32, 32]	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[-600, 600]	0
$F_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_i) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m x_i > a \\ 0 \quad -a < x_i < a \\ k(-x_i - a)^m x_i < -a \end{cases}$	30	[-50, 50]	0
$F_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	[-50, 50]	0

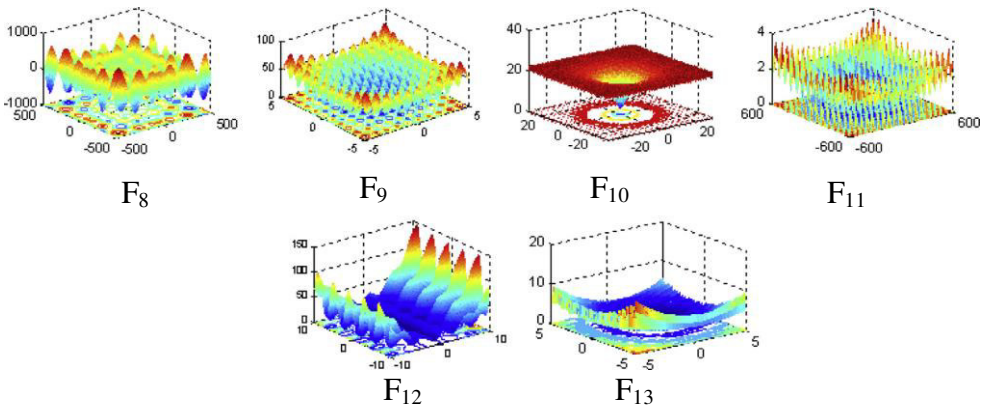

Figura 4.18- Versões 2D das funções *benchmarks multimodais*.

4.1.3 Funções teste *multimodais* de dimensão fixa

Tabela 4.4- Funções *benchmark multimodais* de dimensão fixa.

Função	*Dim*	Range	f_{min}
$F_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2	[-65,65]	1
$F_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	[-5,5]	0.00030
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[-5,5]	-1.0316
$F_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	[-5,5]	0.398
$F_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)$	2	[-2,2]	3
$F_{19}(x) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	3	[1,3]	-3.86
$F_{20}(x) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	6	[0,1]	-3.32
$F_{21}(x) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.1532
$F_{22}(x) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.4028
$F_{23}(x) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	-10.5363

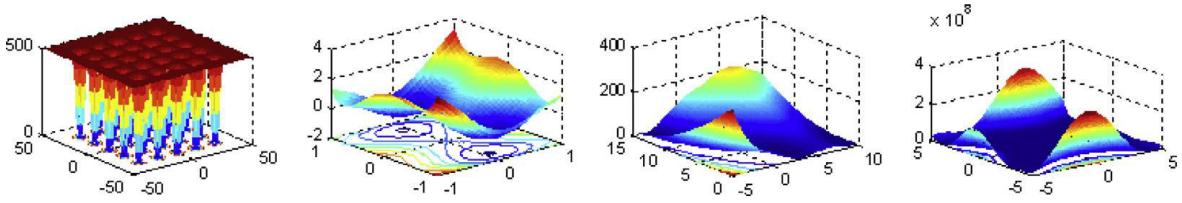

Figura 4.19- Versões 2-D de funções *benchmarks multimodais* de dimensão fixa.

4.3 Otimização das funções teste com o método do gafanhoto-do-deserto

As funções teste utilizadas são funções de minimização e podem ser divididas em três grupos: *unimodais*, *multimodais* e multimodais de dimensão fixa (ver [75] para mais detalhes das funções).

Com o objetivo de avaliar o desempenho do algoritmo proposto, foram selecionadas 23 funções teste, ou *benchmark,* para explorar a capacidade do algoritmo.

Todas as funções são aplicadas a problemas de minimização. Este conjunto de funções foi escolhido também em função da disponibilidade de resultados encontrados na literatura. Os resultados obtidos pelo LSO são comparados com outros tipos de algoritmos de inteligência de enxames: PSO [4], GA[75], ACO [76] e CS [77] (Tabelas 4.5-4.10).

Como índices de comparação foram utilizados o desvio padrão, a média dos valores encontrados pelos algoritmos, o custo de CPU e memória. Todos os algoritmos foram executados 30 vezes. Para execução dos algoritmos foi utilizado o MATLAB, nos índices de comparação para coletar os valores de memória e CPU foram utilizados as funções *memory* e *cputime* respectivamente, de forma que a função *memory* retorna o valor de memória utilizado pelo MATLAB, o arquivo temporário mais o valor usado na função, em MB[1], a função *cputime* retorna o valor em segundos da execução na CPU.

Na Tabela 4.5 são apresentados os resultados das funções *benchmark unimodais*. O LSO proposto consegue fornecer resultados muito competitivos e inclusive este algoritmo supera todos os outros nas funções F_3, F_4, F_5 e F_7. Nota-se que as funções *unimodais* são adequadas para a avaliação comparativa de exploração. Por este motivo, os resultados mostram o desempenho superior do LSO em termos de exploração do ótimo. Isto é devido aos operadores de exploração modelados matematicamente. Na Tabela 4.6 são apresentados os valores dos custos de memória e CPU na execução dessas funções, nota-se que LSO consegue superar o processamento na CPU sobre os GA, ACO e CS, perdendo apenas para o PSO, no requisito memória ele consome 1 MB a mais que os demais algoritmos, superando apenas o ACO e o CS na função F_7.

Oposto às funções *unimodais*, as funções *multimodais* têm muitos ótimos locais com o número crescente exponencialmente com a dimensão. Isto as torna adequadas para

[1] Vale ressaltar que o MATLAB não possui uma função específica que retorne um valor exato do uso da memória em um arquivo individual. Este trabalho foi utilizado esta IDE (MATLAB), então considera-se o valor dado em MB tanto para o uso da IDE como das funções executadas.

medir a capacidade de exploração de um algoritmo. Conforme os resultados das Tabelas 4.7 e 4.9, o LSO consegue fornecer resultados muito competitivos nas funções de referência multimodais. Este algoritmo supera o GA, o CS e o PSO na maioria das funções multimodais. Estes resultados mostram que o algoritmo LSO é superior em termos de exploração. Nas Tabelas 4.8 e 4.10, são apresentados os valores dos custos de memória e CPU na execução das multimodais, nota-se que LSO consegue superar o processamento na CPU sobre os GA, ACO e CS, perdendo apenas para o PSO, no requisito memória ele consome 1 MB a mais que os demais algoritmos, mas possui um desempenho melhor na CPU.

Por fim, observando as curvas de convergência mostradas nas Figuras 4.20 - 4.23 percebe-se que o LSO supera o PSO, GA, ACO e o CS para a maioria das funções teste. Na Figura 4.20, funções *unimodais*, é notório que o LSO consegue superar todos os algoritmos nas funções F_1, F_3, F_4 e F_7, na F_2 ele consegue superar o GA, ACO e o CS, na F_6 supera o PSO e o GA e na F_5 se iguala aos valores do GA, ACO e CS. Para as funções *multimodais*, Figura 4.21, na função F_{10} o LSO supera todos os algoritmos, na F_8 consegue superar o ACO, nas F_9 e F_{11} o LSO consegue encontrar o mínimo global junto com o ACO e nas F_{12} e F_{13} supera o GA, PSO e o CS. Nas Figuras 4.22 e 4.23 o LSO se iguala a todos os algoritmos em todas as funções. Desta maneira é vista um desempenho superior do LSO comparado às demais técnicas existentes.

Vale ressaltar, que o LSO é um método de otimização que consegue alcançar resultados tão bons quanto os de técnicas já existentes na literatura, conforme descrito. Porém, não se pode afirmar que o LSO seja melhor do que todos os demais [78].

Tabela 4.5- Resultados do LSO nas funções *benchmarks unimodais*.

F	LSO		PSO		GA		ACO		CS	
	ave	**std**	**ave**	**std**	**ave**	**std**	**ave**	**std**	**ave**	**std**
F_1	7.129e-21	1.7234e-20	3.9664e-07	6.1679e-07	0.01175	0.0077662	9.6358e-21	8.0112e-21	4.0239e-10	1.4023e-09
F_2	5.7601e-11	1.6949e-10	4.7533e-05	4.2783e-05	0.86525	1.1951	2.1099e-15	3.0315e-15	2.3957e-07	2.202e-07
F_3	5.0588e-05	4.0432e-05	9.7477	4.2048	5575	1937.3	28149	6542	9786.2	4216
F_4	0.00057653	0.00027204	0.85913	0.15687	1.0144	0.18151	8.9873	4.1529	18.165	3.6037
F_5	27.096	0.09749	51.576	35.566	591.27	909.28	36.968	26.85	52.784	47.529
F_6	3.5294e-07	5.5631e-07	2.6821e-07	3.6361e-07	0.01189	0.0091173	1.0298e-20	1.4861e-20	3.1358e-10	9.1025e-10
F_7	0.0018187	0.0010675	14.303	10.371	0.014964	0.0066933	0.021135	0.0060806	0.068039	0.031011

Tabela 4.6- Valores de memória e CPU do LSO nas funções *benchmarks unimodais*.

F	LSO		PSO		GA		ACO		CS	
	Memória MATLAB (MB)	**CPU (segundos)**	**Memória MATLAB (MB)**	**CPU (segundos)**	**Memória MATLAB (MB)**	**CPU (segundos)**	**Memória MATLAB (MB)**	**CPU (segundos)**	**Memória MATLAB (MB)**	**CPU (segundos)**
F_1	873	2.9058	872	1.6198	872	8.2592	872	1509.3	872	5.0076
F_2	873	3.1939	872	1.8107	872	8.5587	872	3650.3	872	5.4195
F_3	873	11.323	872	9.4635	872	19.571	872	6618.2	872	21.442
F_4	873	3.8012	872	2.4373	872	9.7475	872	1249.8	872	6.0097
F_5	873	3.9167	872	2.5797	872	9.943	871	3675	871	6.8292
F_6	872	3.0769	871	1.7462	871	9.0845	871	6036.6	871	5.473
F_7	872	4.0181	871	2.7035	871	10.287	875	8424.4	875	7.2494

Tabela 4.7- Resultados do LSO nas funções *benchmarks multimodais*.

F	LSO		PSO		GA		ACO		CS	
	ave	**std**	**ave**	**std**	**ave**	**std**	**ave**	**std**	**ave**	**std**
F_8	-5683.2	433.78	-6889.6	781.75	-10754	444.38	-5110.4	336.48	-7605.5	907.24
F_9	1.7251	3.8205	57.83	20.733	1.492	0.65514	218.45	12.747	88.191	21.916
F_{10}	1.0975e-11	9.2095e-12	0.00026751	0.00017414	0.43071	0.35358	14.095	9.2276	2.6163e-05	4.8314e-05
F_{11}	0.0091747	0.012312	0.013947	0.011517	0.053738	0.039508	0.013807	0.040968	0.0092999	0.013846
F_{12}	2.6344e-08	6.6302e-08	2.7742e-09	5.3456e-09	0.020754	0.049353	2.9024e-16	6.0539e-16	1.9773	1.875
F_{13}	3.1589e-07	8.9863e-07	0.00037791	0.0019706	0.004006	0.0052112	9.0517e-15	2.2192e-14	1.6985	2.5968

Tabela 4.8- Valores de memória e CPU do LSO nas funções *benchmarks multimodais*.

F	LSO		PSO		GA		ACO		CS	
	Memória MATLAB (MB)	**CPU (segundos)**	**Memória MATLAB (MB)**	**CPU (segundos)**	**Memória MATLAB (MB)**	**CPU (segundos)**	**Memória MATLAB (MB)**	**CPU (segundos)**	**Memória MATLAB (MB)**	**CPU (segundos)**
F_8	876	3.4414	875	2.0733	875	9.6305	875	10864	875	6.1553
F_9	876	3.6125	875	2.262	875	9.4537	875	13242	875	6.2593
F_{10}	876	4.2375	875	2.8403	875	10.321	875	15704	875	6.8588
F_{11}	876	3.7919	875	2.4139	875	10.234	875	18188	875	6.7132
F_{12}	876	6.3482	875	5.0648	875	13.107	875	20843	875	12.069
F_{13}	876	7.0928	875	5.575	875	15.175	875	23904	875	13.517

Tabela 4.9- Resultados do LSO nas funções *benchmarks multimodais* de dimensão fixa.

F	LSO		PSO		GA		ACO		CS	
	ave	**std**	**ave**	**std**	**ave**	**std**	**ave**	**std**	**ave**	**std**
F_{14}	0.998	0	3.956	2.4612	0.998	0 + 1.825e-08i	1.1624	0.88511	0.998	0 + 2.1073e-08i
F_{15}	0.00035439	0.00023161	0.0046464	0.0071591	0.0014636	0.0035243	0.0016754	0.0049973	0.00037256	6.2617e-05
F_{16}	-1.0316	2.581e-08	-1.0316	4.2147e-08	-1.0316	4.2147e-08	-1.0316	4.2147e-08	-1.0316	4.2147e-08
F_{17}	0.39789	3.7991e-08	0.39789	5.2684e-09	0.39789	3.3224e-07	0.39789	5.2684e-09	0.39789	5.2684e-09
F_{18}	3	0 + 4.2147e-08i	3	1.1151e-07	3	1.2644e-07	3	1.3328e-07	3	9.4243e-08
F_{19}	-3.8628	1.1921e-07	-3.2615	1.1947	-3.8628	8.4294e-08	-3.8628	1.0324e-07	-3.8628	1.0324e-07
F_{20}	-3.2823	0.056119	-1.9985	0.67231	-3.2903	0.052577	-3.2784	0.057294	-3.3061	0.040416
F_{21}	-6.2463	2.1554	-3.3857	2.0447	-8.3188	3.0904	-6.6658	3.7282	-10.153	3.7697e-07
F_{22}	-7.391	2.6339	-3.0991	2.1286	-7.5375	3.5228	-8.3749	3.1124	-10.403	2.6656e-07
F_{23}	-6.9311	2.5493	-3.9148	2.309	-7.8031	3.6038	-8.6	3.2545	-10.536	0 + 2.92e-07i

Tabela 4.10- Valores de memória e CPU do LSO nas funções *benchmarks multimodais* com dimensão fixa.

F	LSO		PSO		GA		ACO		CS	
	Memória MATLAB (MB)	**CPU (segundos)**	**Memória MATLAB (MB)**	**CPU (segundos)**	**Memória MATLAB (MB)**	**CPU (segundos)**	**Memória MATLAB (MB)**	**CPU (segundos)**	**Memória MATLAB (MB)**	**CPU (segundos)**
F_{14}	876	17.318	875	15.441	875	27.767	875	27328	875	32.428
F_{15}	876	3.614	875	1.9318	875	10.332	875	29628	875	5.8766
F_{16}	876	2.6115	875	1.1638	875	7.8006	875	34010	875	4.004
F_{17}	876	2.3801	875	1.0369	875	7.1485	875	31240	875	3.7003
F_{18}	876	2.8512	874	1.2745	874	8.5437	874	31919	874	4.5121
F_{19}	876	3.7685	874	2.3691	874	9.2061	874	34979	874	6.5089
F_{20}	876	3.8662	874	2.3951	874	9.6591	874	36032	874	6.628
F_{21}	876	4.8324	874	3.3124	874	10.742	875	37210	875	7.734
F_{22}	876	5.5126	875	4.0004	875	11.882	875	38466	875	9.0559
F_{23}	876	6.2593	875	4.691	875	12.717	875	39896	875	10.415

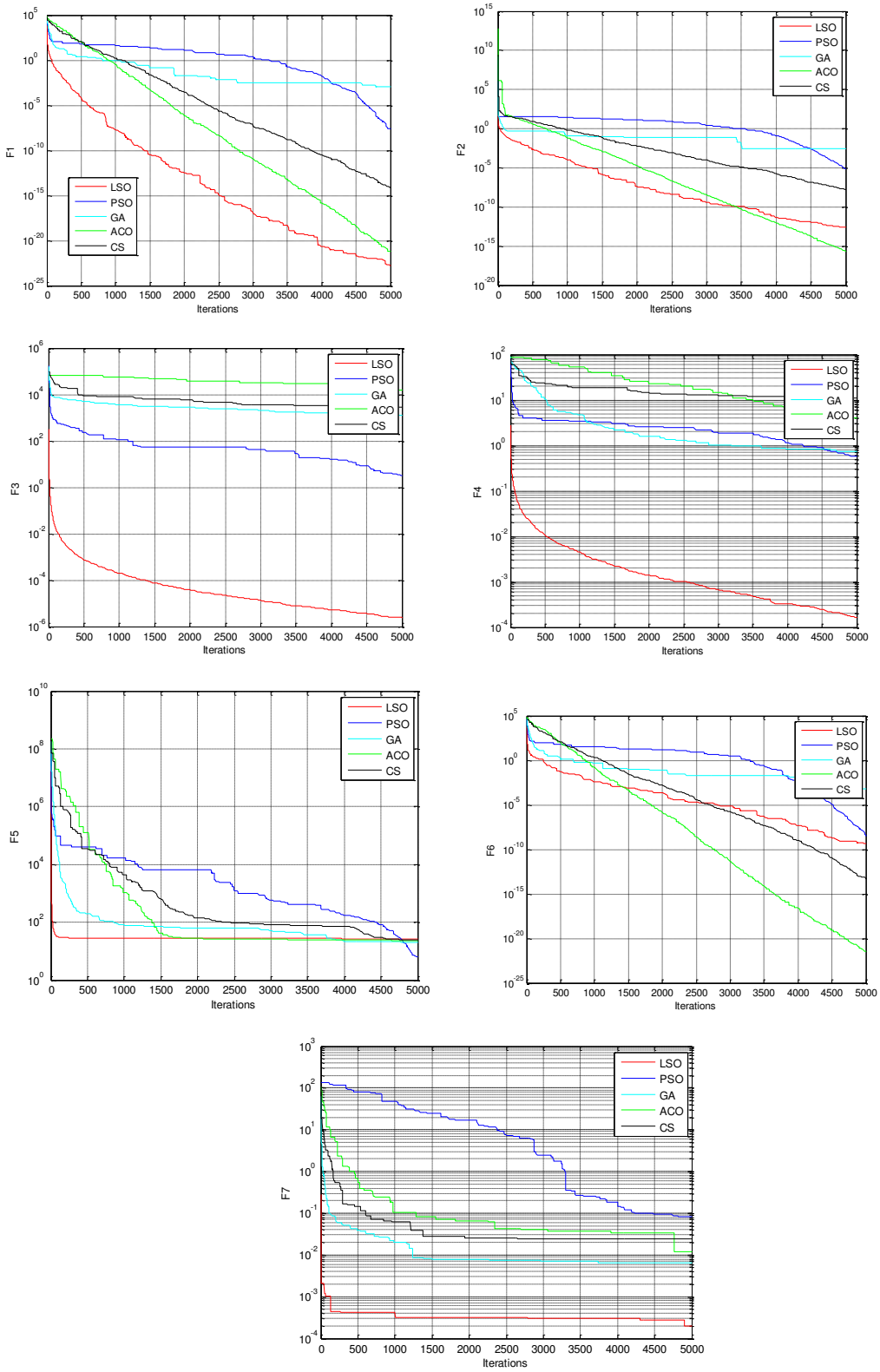

Figura 4.20- Curva de convergência do LSO nas funções teste unimodais F1-F7.

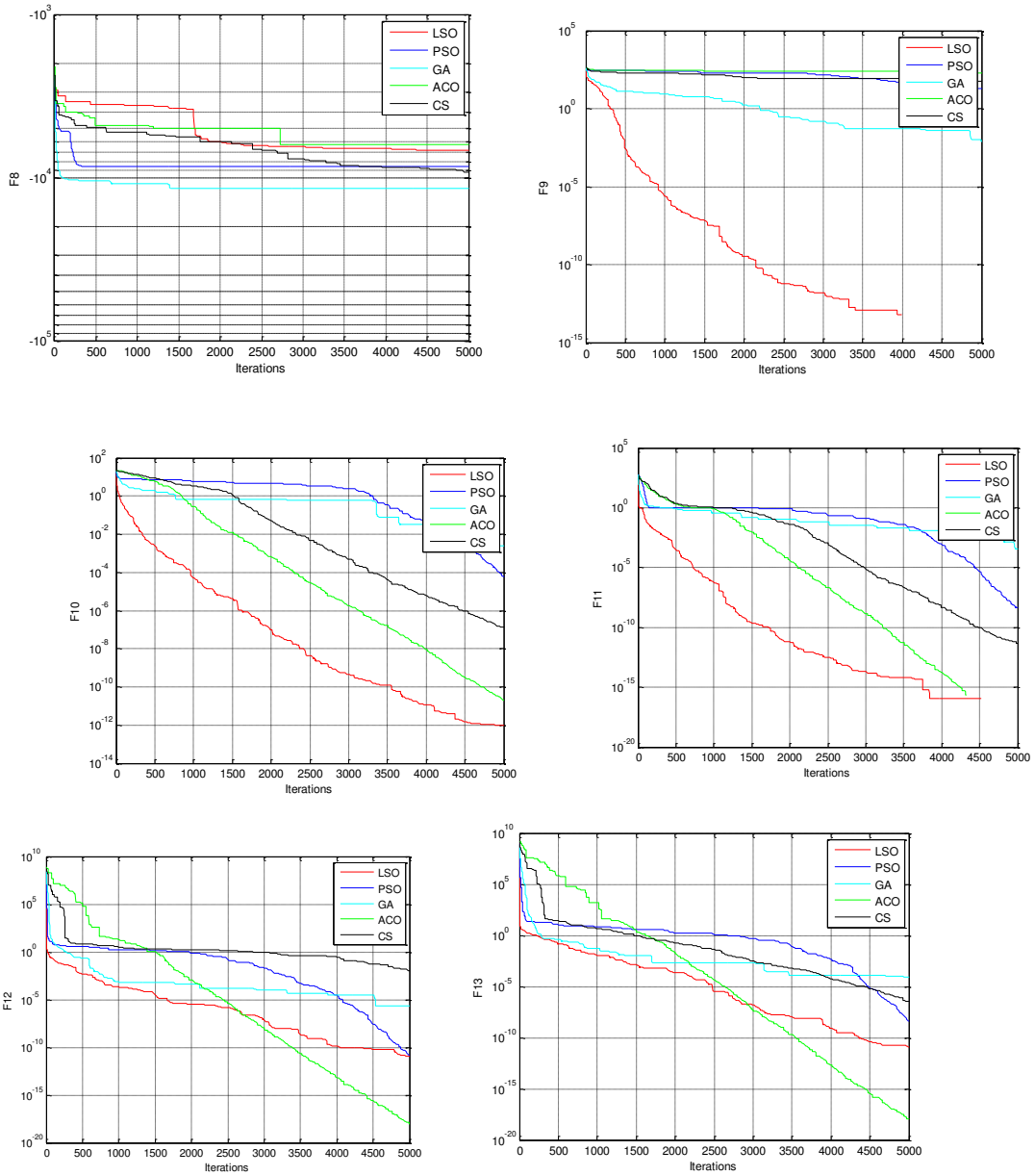

Figura 4.21- Curva de convergência do LSO nas funções teste multimodais F8-F13.

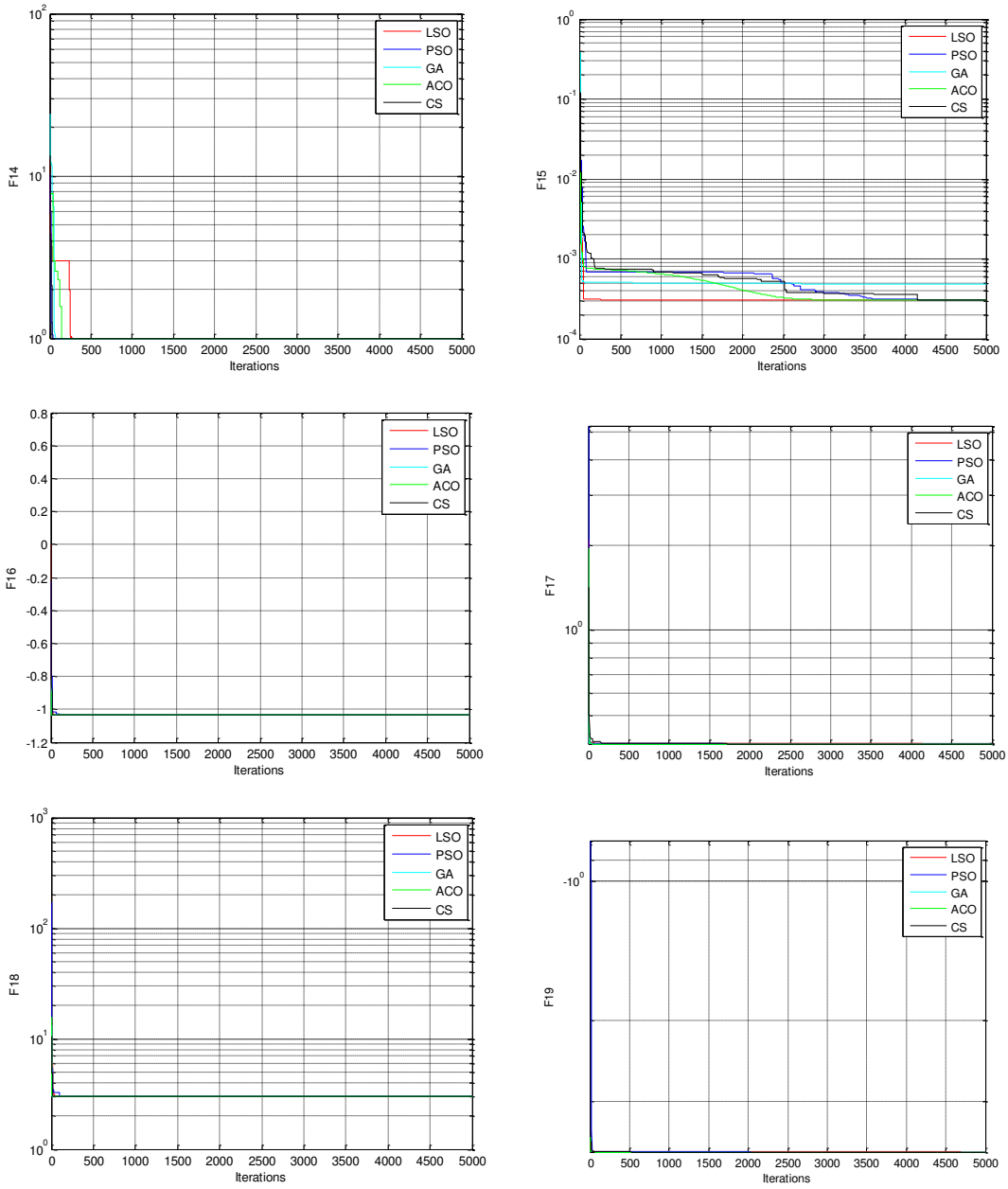

Figura 4.22- Curva de convergência do LSO nas funções teste multimodais de dimensão fixa F14-F19.

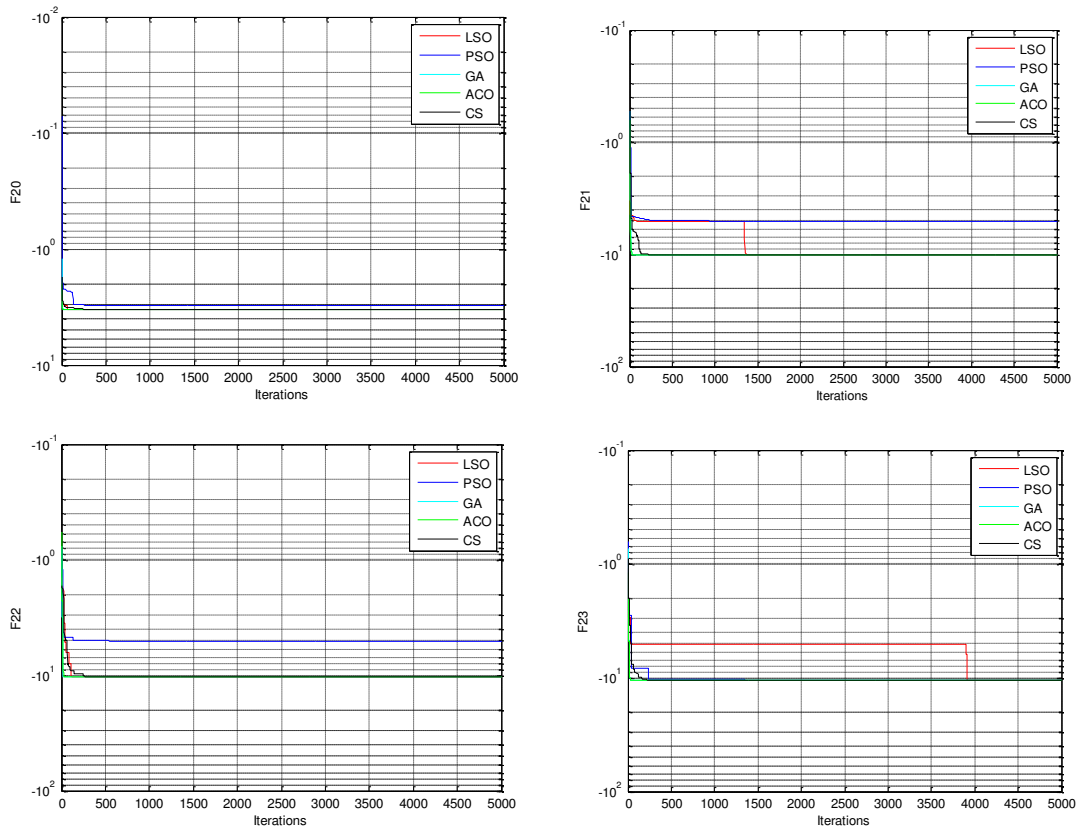

Figura 4.23- Curva de convergência do LSO nas funções teste multimodais de dimensão fixa F20-F23.

4.4 Modelos de aplicações do LSO em sistemas elétricos

Nesta subseção o LSO será aplicado na solução do problema do despacho econômico de geradores termelétricos em sistemas de energia elétrica.

4.4.1 Operação econômica de sistemas de energia elétrica

De forma geral, o problema do despacho econômico de cargas elétricas é um estudo do uso ótimo de unidades geradoras presentes no sistema elétrico [79]. No qual, se tem por objetivo a minimização do custo de produção de energia elétrica, de modo que satisfaça as condições de operação do sistema em estudo.

Para a montagem do cenário [79], deve-se considerar um sistema de potência que seja formado por N unidades geradoras térmicas que alimentam N_L cargas conectadas a

barras da rede de eletricidade. Seja P_{L_i} a potência da i-ésima carga, por tanto, a carga total deste sistema será dada por:

$$P_L = \sum_{i=1}^{N_L} P_{L_i} \tag{4.6}$$

Para essa subseção, será utilizado um modelo simplificado de rede elétrica, de modo que a carga total em P_L e as unidades geradoras estão conectadas a uma única barra, como mostrado na Figura 4.24. Deve-se desprezar inicialmente as perdas de transmissão.

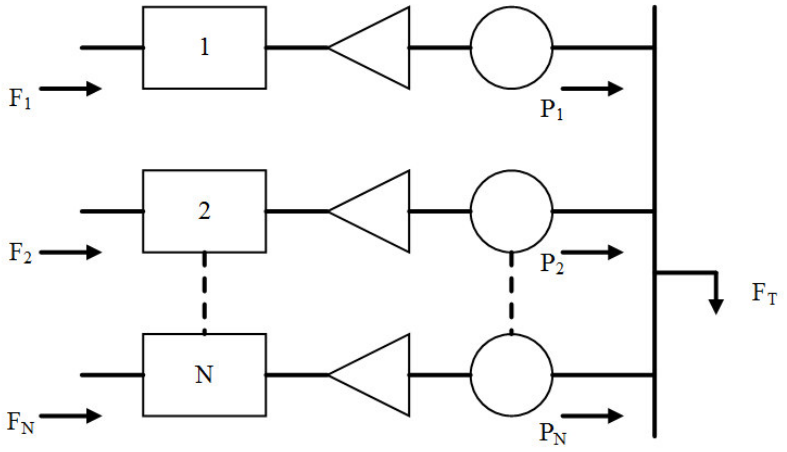

Figura 4.24- Representação das unidades térmicas [86].

A função de custo da i-ésima unidade geradora é representada por $F_i(P_i)$, e destacada em \$/h, de modo que P_i seja a potência gerada pela unidade i. A função de custo total do sistema será representada por (4.7):

$$F_T(P_1, P_2, \dots, P_N) = \sum_{i=1}^{N} F_i\,(P_i) \tag{4.7}$$

Dessa maneira, F_T é separável por unidade geradora.

As perdas de transmissão devem ser desprezadas, para que um despacho viável das unidades geradoras satisfaça à equação de balanço de potência (4.8):

$$\sum_{i=1}^{N} P_i = P_L \tag{4.8}$$

Ademais, cada uma das unidades geradora está exposta em seus limites máximo e mínimo para cada geração, isto é (4.9):

$$P_i \leq P_i \leq \overline{P_i}, i = 1, \dots, N \tag{4.9}$$

Onde P_i e $\overline{P_i}$ são nesta ordem os limites mínimo e máximo de geração para a unidade i.

O LSO, irá resolver o seguinte problema de despacho econômico, descrito por [79]. Suponha-se que é desejado determinar o ponto de operação econômica para essas três unidades geradoras a seguir, ao entregar um total de 850MW:

Unidade 1. Unidade de vapor a carvão:

- Saída máxima = 600MW;

- Potência mínima = 150MW;

- Curva de entrada-saída:

$$H_1\left(\frac{MBtu}{h}\right) = 510{,}0 + 7{,}2P_1 + 0{,}00142P_1^2$$

Unidade 2. Unidade de vapor a óleo:

- Saída máxima = 400MW;

- Potência mínima = 100MW;

- Curva de entrada-saída:

$$H_2\left(\frac{MBtu}{h}\right) = 310{,}0 + 7{,}85P_2 + 0{,}00194P_2^2$$

Unidade 3. Unidade de vapor a óleo:

- Saída máxima = 200MW;

- Potência mínima = 50MW;

- Curva de entrada-saída:

$$H_3\left(\frac{MBtu}{h}\right) = 78{,}0 + 7{,}97P_3 + 0{,}00482P_3^2$$

Antes que este problema possa ser resolvido, o custo do combustível de cada unidade deve ser especificado. Assim, serão considerados os seguintes custos de combustível a estar em vigor:

Unidade 1: Custo de combustível = 1,1 $\$/MBtu$

Unidade 2: Custo de combustível = 1,0 $\$/MBtu$

Unidade 3: Custo de combustível = 1,0 $\$/MBtu$

Então:

$$F_1(P_1) = H_1(P_1) \times 1{,}1 = (561 + 7{,}92P_1 + 0{,}001562P_1^2) \times 1{,}1\$/h$$

$$F_2(P_2) = H_2(P_2) \times 1{,}0 = (310 + 7{,}85P_2 + 0{,}00194P_2^2) \times 1{,}0\$/h$$

$$F_3(P_3) = H_3(P_3) \times 1{,}0 = (78 + 7{,}97P_3 + 0{,}00482P_3^2) \times 1{,}1\$/h$$

Logo, o problema de otimização é formulado assim:

$$\begin{cases} Min\ (F_1 + F_2 + F_3) \\ sujeito\ a: \\ P_1 + P_2 + P_3 = 850 \\ 150 \leq P_1 \leq 600 \\ 100 \leq P_2 \leq 400 \\ 50 \leq P_3 \leq 200 \end{cases}$$

Com o uso dos algoritmos LSO, PSO,GA, ACO e CS e foram obtidos os resultados apresentados na Tabela 4.11. Cada algoritmo foi executado 30, 50 e 100 vezes com uma população de 30 e com 2000 iterações, para ter uma certeza do comportamento do LSO.

Tabela 4.11- Resultados do LSO para o problema do despacho econômico.

Algoritmo	30 execuções				50 execuções				100 execuções			
	ave	**std**	**Memória MATLAB (MB)**	**CPU (segundos)**	**ave**	**std**	**Memória MATLAB (MB)**	**CPU (segundos)**	**ave**	**std**	**Memória MATLAB (MB)**	**CPU (segundos)**
LSO	8196.5	1.645	873	1.0858	8196.9	2.4143	873	1.106	8198	3.6158	873	1.1176
PSO	8203.3	13.154	873	0.5174	8200.8	7.0192	873	0.5198	8203	10.065	873	0.51933
GA	8219.1	26.738	873	3.1361	8210.3	15.412	873	3.1094	8206.5	12.983	873	3.1217
ACO	8195.9	3.5621	873	42801	8195	0.65336	873	43202	8195.4	2.755	873	43950
CS	8195.3	0.72376	873	1.7332	8195.3	0.62634	873	1.7382	8195.3	0.66888	873	1.7505

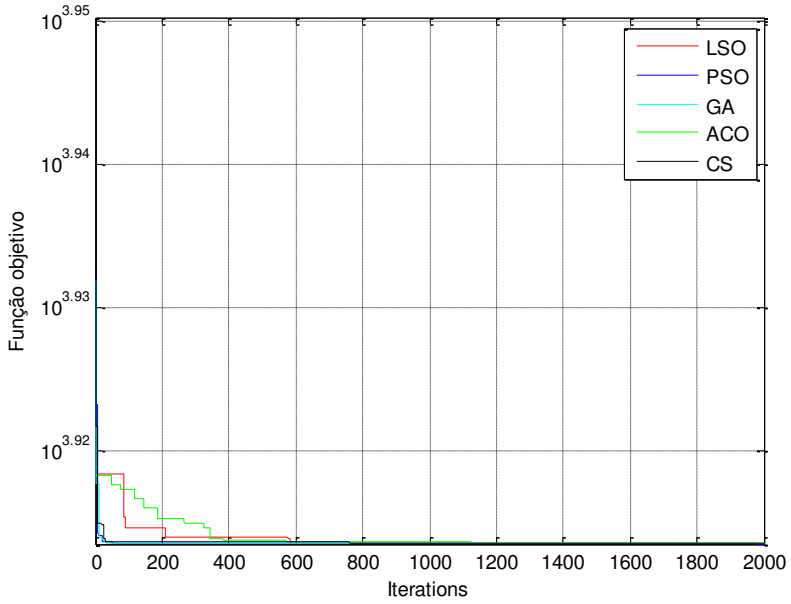

Figura 4.25- Curva de convergência do LSO no problema do despacho econômico com 30 execuções.

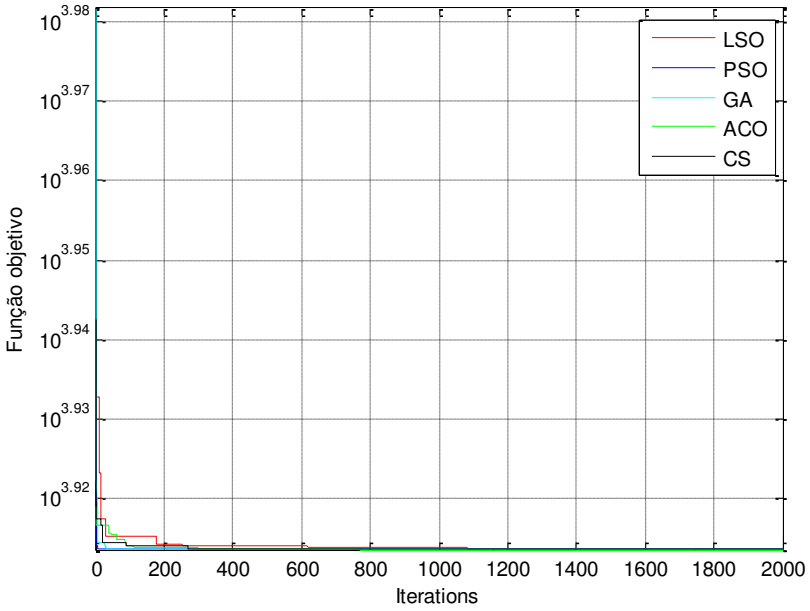

Figura 4.26- Curva de convergência do LSO no problema do despacho econômico com 50 execuções.

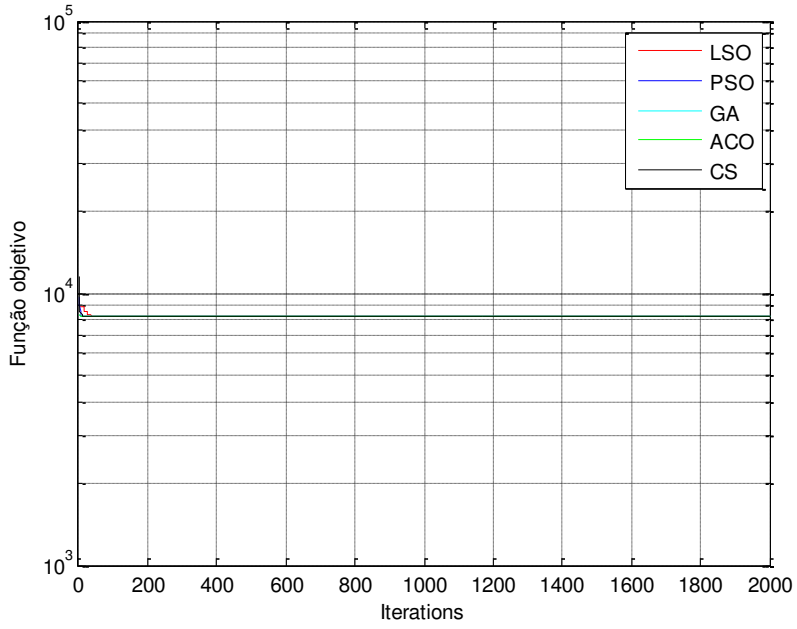

Figura 4.27- Curva de convergência do LSO no problema do despacho econômico com 100 execuções.

Também foram analisadas as curvas de convergência dos algoritmos para a solução do problema do despacho econômico representados nas Figuras 4.25-4.27 para 30, 50 e 100 execuções, respectivamente. O LSO com 30, 50 e 100 execuções consegue superar o GA e o PSO perdendo apenas para o ACO e GS, porém no tempo de processamento (CPU) o LSO consegue ser mais rápido que o GA, ACO e CS perdendo apenas para o PSO (30 e 100 execuções), na memória do sistema o LSO possui um valor igual aos demais algoritmos. Na curva de convergência é possível notar que o LSO consegue se igualar aos demais algoritmos, tornando-se um algoritmo competitivo dentre os algoritmos bioinspirados na natureza.

4.5 Modelos de aplicações prospectivas do LSO

Nesta subseção alguns modelos de aplicações prospectivas do LSO serão abordados, de forma que o algoritmo proposto possa trabalhar com sistemas computacionais, otimização combinatória e redes de comunicação.

4.5.1 Sistemas computacionais: Segurança computacional

No presente cenário atual é frequente o uso de redes computacionais que interligam as pessoas e as máquinas, devido à união das telecomunicações e da informática que muito contribuíram para que isto ocorresse.

Por causa desta junção máquinas trabalham melhores, auxiliando organizações a executarem suas tarefas com maior competência, pois, não só as organizações estão sofrendo mudanças, mas a própria sociedade está vivenciando esse novo tempo o qual muitos autores e estudiosos consideram a Era da Informação.

A segurança é um fator primordial quando se trabalha em redes, pois através dela podem-se proteger os dados importantes e prevenir de fraudes ou roubos. Conforme cita Tanenbaum [80], a segurança é um assunto abrangente e inclui inúmeros tipos de problema e se preocupa em garantir que pessoas com más intenções não leiam ou modifiquem secretamente mensagens enviadas a outros destinatários buscando saber se uma mensagem verdadeira é falsa ou não.

Neste contexto de garantir a segurança de uma rede computacional, métodos inteligentes são usados para garantir a autenticidade de dados em uma rede [81-83]. O método proposto nesta pesquisa (LSO) poderá auxiliar na proteção de redes computacionais otimizando informações e/ou protegendo dados de possíveis ataques em

servidores. Assim, a seguir será proposta um modelo de arquitetura, Figura 4.28, que busca resolver possíveis ataques em uma rede computacional.

Para a montagem do cenário, considerar-se-á uma rede de computadores de computadores interligados que estão trocando informações (dados) a um servidor na rede. Este servidor é responsável pelo serviço entre cliente e servidor, pela análise dos dados, e o envio dos mesmos, suponha-se que este servidor está sob um ataque, como por exemplo, um ataque DDoS (*Distributed Denial of Service*) no qual este tipo de ataque se caracteriza em derrubar os serviços do servidor com inúmeros processos até que o mesmo não suporte a demanda e o serviço pare de funcionar. Para se trabalhar com o comportamento do gafanhoto na solução deste problema, será dividido em quatro etapas, a identificação, a verificação, a análise e por fim a solução (ou reparo). No qual, haverá um agente gerenciador que trabalhará com o comportamento e análise de dados.

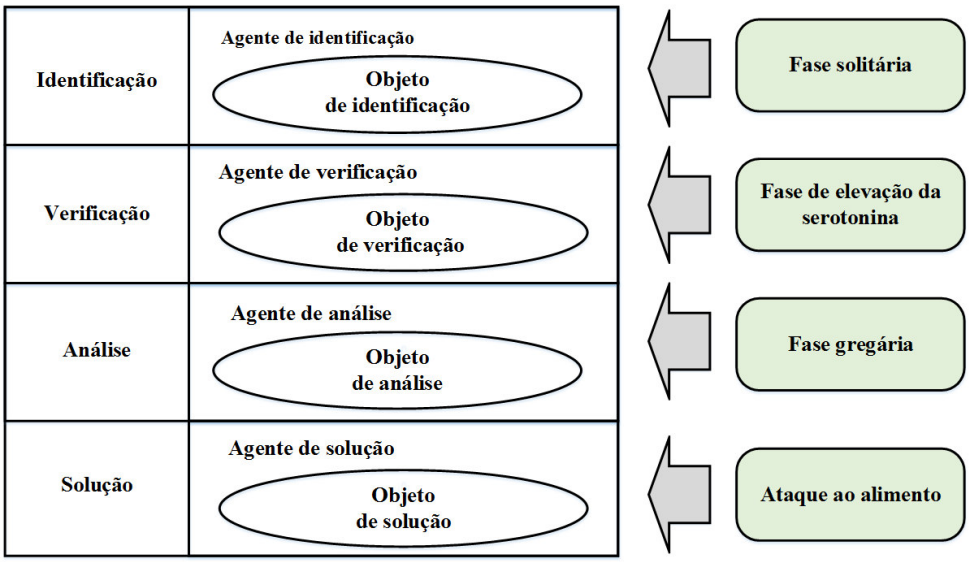

Figura 4.28– Arquitetura de um modelo de segurança baseado no comportamento do gafanhoto-do-deserto.

Identificação: Nesta etapa, o agente identificador irá identificar quais máquinas que estão sofrendo o ataque. Quais máquinas estão fora do padrão normal, seja devido ao elevado número de processos na CPU, ou devido a um elevado número de processos na memória, ou qualquer fator que esteja fora do que se considera normal. Esta etapa no comportamento do gafanhoto consiste nos insetos solitários.

Verificação: Nesta etapa, o agente de verificação, irá buscar (solicitar acesso) na rede outras máquinas que poderiam ajudar no processamento dos dados que estão sobrepondo a máquina que está sofrendo o ataque. A verificação consiste em quantidades de processos nas CPUs dos clientes, e quantidade de memória disponível. Esta etapa no comportamento do gafanhoto consiste no estímulo do aumento da serotonina.

Análise: Nesta etapa, o agente de análise irá analisar as máquinas solicitadas, e enviará pacotes testes para se verificar se o computador solicitado suportará ou não os processos conflitantes. A análise é uma etapa de análise de erros, e busca de uma nova solução (uma nova máquina na rede). Esta etapa no comportamento do gafanhoto consiste na busca e formação de enxames do gafanhoto.

Solução: Por fim, nesta etapa, o agente irá distribuir os pacotes conflitantes para as máquinas clientes que atenderão a demanda, e poderão ajudar no processamento. Evitando assim, a sobrecarga no servidor, e auxiliando para que o servidor continue a funcionar. Esta etapa no comportamento do gafanhoto consiste no ataque dos insetos ao alimento.

4.5.2 Sistemas computacionais: Anomalias de dados na nuvem

A detecção de anomalias se torna essencial para a análise de quesitos de segurança, e levando isso para a computação em nuvem o torna ainda mais importante, pois ela possibilita assegurar a segurança e confiabilidade dos dados que trafegam em sua rede.

As anomalias podem ser identificadas a partir de mudanças acentuadas no fluxo da rede, comparando o mesmo com o fluxo normal que a mesma teria [84].

A inteligência de enxame dos gafanhotos possui características peculiares que serão usadas para o processo do modelo proposto, baseado nessas características que propõe-se uma metodologia de detecção de anomalias da rede na nuvem baseado na inteligência do enxame do gafanhoto do deserto para garantir a segurança na rede.

Devido à possibilidade de busca individual dos gafanhotos, e sua capacidade de agregação em forma de enxame, quando avistado um potencial alvo (o alimento); sua habilidade se apresenta como uma forma de busca de fluxo normal de dados na rede que pode trazer resultados significativos. Com a implementação dessa característica é possível verificar toda a estrutura da rede na nuvem de forma silenciosa, ou seja, de forma a não realizar um consumo grande de recursos, como CPU, memória, dentre outros mais.

Na Figura 4.29 detalha-se a arquitetura do modelo proposto, sendo dividido o processo em etapas e que se assemelha à formação do enxame dos gafanhotos aplicado ao volume de dados que trafega na rede da nuvem. O modelo proposto é dividido em quatro etapas, como já mencionadas, essas etapas são os processos da arquitetura, e cada uma possui sua responsabilidade. A seguir a descrição de cada etapa.

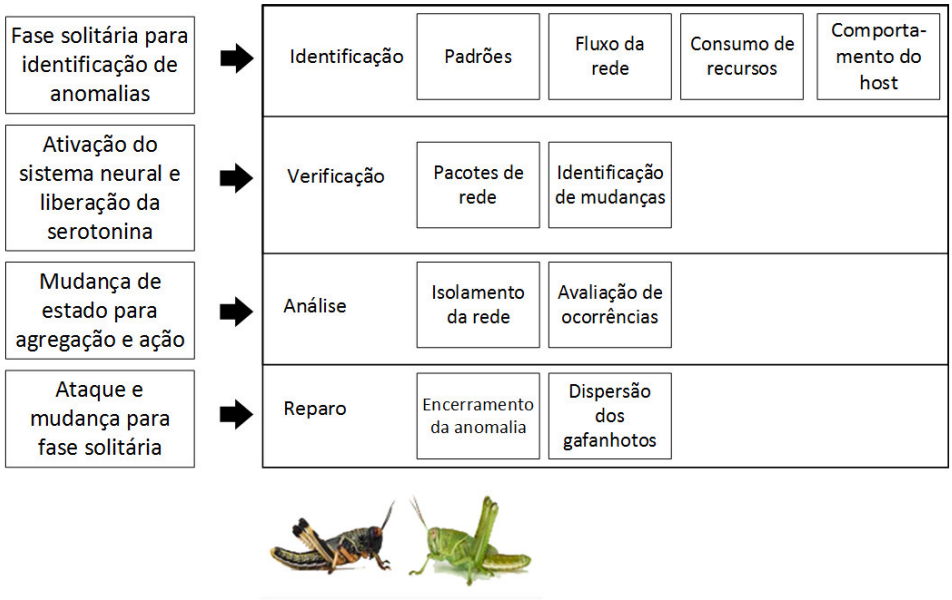

Figura 4.29- Arquitetura do modelo de segurança de anomalias em rede baseado no comportamento do gafanhoto-do-deserto.

Identificação: O processo de identificação é baseado na vida do gafanhoto do deserto, pode ser caracterizado pela busca solitária por alimento se comparado ao inseto. Nele os objetos de identificação vasculham a rede atrás de informações sobre os padrões, o fluxo de rede, o consumo de recursos e o comportamento do *host*, a fim de encontrar possíveis variações no seu processo comum. Após identificar esse desvio de padrões segue o processo de verificação.

Verificação: Neste processo são analisados os pacotes dos dados a fim de identificar mudanças, principalmente no teor do pacote ou se o mesmo está corrompido depois de identificado algo suspeito. Na verificação usa-se a técnica de *sniffing* para verificar o conteúdo dos dados. Caso identificada a anomalia, passa-se para o processo de análise. A análise é realizada levando em conta o comportamento do gafanhoto e corresponde à etapa de estímulo do aumento da serotonina em outros insetos na região, onde após o estímulo os gafanhotos se agregam e o enxame deles parte para o ataque.

Análise: Processo que ocorre quando se é verificada a anomalia da rede na nuvem, nela, baseado no que se é determinado, o processo pode correr da seguinte forma, ou promove o isolamento da rede até a realização do extermínio da anomalia (caso de tentativa de intrusão), ou realiza apenas a avaliação da ocorrência, a fim de determinar o problema e a possível solução (dados corrompidos, por exemplo). Então se passa para o processo final. Em comparação ao gafanhoto, esse processo corresponde à fase de agregação, onde um número considerado de gafanhotos já está pronto para o ataque ao alimento.

Reparo: Essa função do objeto está associada ao uso de uma base de conhecimentos para a solução do ocorrido, como seria o caso do bloqueio do tráfego de dados caso seja detectada uma possível intrusão, dentre outras mais. Ao concluir o reparo, ocorre a dispersão dos processos, que, se comparado aos gafanhotos, seria como se eles tivessem terminado seu alimento e retornado ao seu estado solitário.

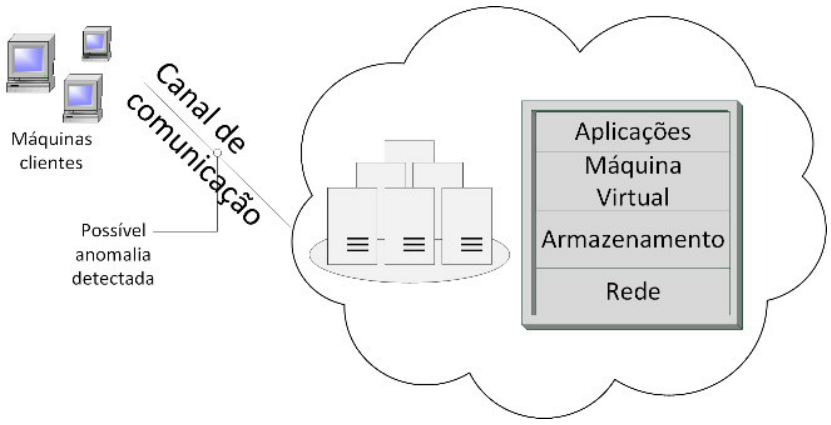

Figura 4.30- Detecção de anomalia pelo modelo do gafanhoto.

Os objetos de identificação (gafanhotos solitários) estão dispersos na rede na forma aleatória, a fim de identificar variações da rede. Na Figura 4.30, representa-se uma possível detecção da anomalia no canal de comunicação da nuvem por parte do modelo do

gafanhoto, no qual ele identifica mudança quanto ao padrão da rede e logo começa a analisar.

O canal de comunicação com o cliente deve estar sempre protegido, no caso, ele deve ter objetos de identificação acompanhando em todo momento os processos na rede. A proposta define a existência de pelo menos um objeto de identificação por rede conectada à nuvem, para realizar isso, protegendo os dados na rede, principalmente nos componentes da aplicação, da máquina virtual, do armazenamento e da rede em si.

4.5.3 Otimização combinatória: Caixeiro viajante

O problema do caixeiro viajante (em inglês, *Traveling Salesman Problem* (TSP)) é um problema que busca determinar a menor rota para percorrer um número de cidades, onde o caixeiro deve passar apenas uma vez por cada um deles e finalmente retornar à cidade de origem. [85]

O modelo do gafanhoto do deserto será adaptado à solução deste problema. Este modelo irá considerar um agente para atuar como o caixeiro viajante e um grupo de agentes que agem com a finalidade de armazenar as posições e os custos de cada rota, que serão chamados aqui de agentes de armazenamento. No exemplo da Figura 4.31 é considerado um conjunto de cidades conectadas de A à J, cada interconexão de duas cidades tem um custo associado.

Identificação: Nesta etapa será identificado o objetivo do caminho. Deve saber em que cidade o caixeiro viajante deve ir. Este é o problema a ser resolvido, as rotas são o tamanho deste problema e as conexões entre as cidades são as iterações. Este passo é representado pela fase solitária do gafanhoto do deserto.

Verificação: Nesta etapa, os agentes de armazenamento irão verificar e calcular as rotas existentes a seguir, e o custo que está em cada iteração entre eles. Este passo representa a fase de aumento da serotonina dos gafanhotos para alterar a fase.

Análise: Nesta fase, os agentes de armazenamento terão a capacidade de separar os melhores custos dos piores. Os agentes de armazenamento manterão as melhores posições e possibilidades das trajetórias de menor custo. Esta fase é representada pelos gafanhotos do deserto mudando de fase solitária para fase de agregação.

Solução: Depois de ter feito as melhores posições de armazenamento com o menor custo, então nesta fase o caixeiro viajante pode ser enviado para a cidade de destino, completando assim o seu objetivo. A Figura 4.31, é um exemplo de TSP e é mostrada a estrutura do modelo proposto para resolver este problema com LS.

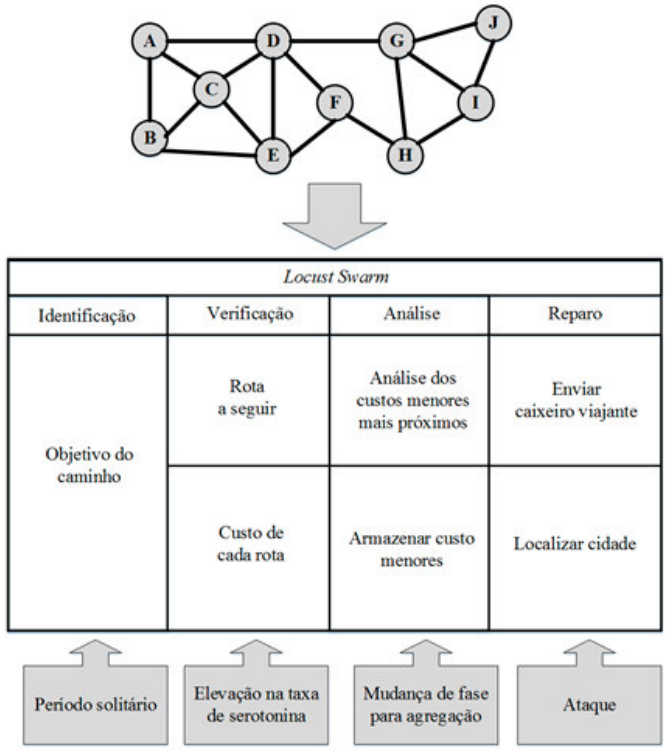

Figura 4.31- Modelo baseado no enxame de gafanhotos para solução do TSP.

4.5.4 Redes de comunicação: Roteamento

O envio da informação se dá através das redes de comunicação. Os padrões de tráfego, nível de carga e topologia, operações comuns na rede, ao longo do tempo podem sofrer variações, porém algumas delas não são previstas, tornando assim o ambiente dinâmico e complexo. O sistema de roteamento, destaque dos processos de transmissão de informação, é responsável por controlar o fluxo de informações percorridas ao longo das redes. Isto torna este sistema responsável também pelo desempenho da rede [86].

O balanceamento de cargas nas redes de comunicação são soluções para o problema do roteamento. A modulação deste modelo, Figura 4.32, para trabalhar com o comportamento do gafanhoto é uma forma de exemplificação do modelo abordado nesta pesquisa.

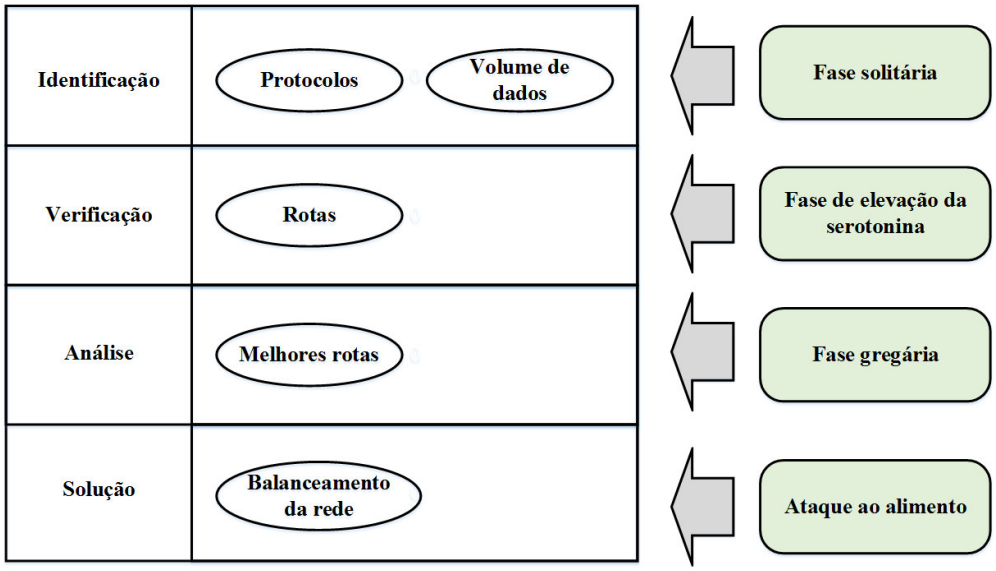

Figura 4.32– Arquitetura de um modelo de roteamento de redes de comunicações baseado no comportamento do gafanhoto-do-deserto.

Identificação: Nesta etapa será identificada a quantidade de informações a serem transmitidas na rede. Devem-se saber quais protocolos e quais volumes de dados que serão

enviados. Este é o problema a ser solucionado em que os dados são a dimensão deste problema e o envio destes dados aos seus clientes são as iterações. Esta etapa representa a fase solitária do gafanhoto-do-deserto.

Verificação: Nesta etapa, os agentes de armazenamentos irão verificar e calcular as rotas em que se deseja otimizar e em quais rotas há um excesso de informações. Esta etapa representa a fase do aumento da serotonina dos gafanhotos para mudar de fase.

Análise: Nesta etapa, os agentes de armazenamentos irão armazenar as melhores rotas de troca de informações entre os computadores dos clientes que deverão requisitar o recebimento de dados enviados. Esta etapa representa a mudança de fase solitária para fase de agregação dos gafanhotos do deserto.

Solução: Após liberado o acesso, os pacotes são enviados nas melhores rotas que foram localizadas, liberando assim o fluxo de informações e balanceando a rede. Finalizando o objetivo que se deseja.

Neste capítulo 4 foram abordados os resultados dos testes feitos com o LSO, assim como as prospectivas de modelos que o LSO poderá servir futuramente. A seguir, no capítulo 5 serão abordadas as conclusões feitas a partir destes resultados apresentados anteriormente.

5 Conclusão

5.1 Conclusões

Nesta pesquisa foi proposto um novo algoritmo de otimização de inteligência de enxames inspirado no comportamento de enxames do gafanhoto-do-deserto com variáveis reais. O método proposto reproduz a formação de nuvens de ataques do gafanhoto do deserto.

No estudo desta pesquisa foram estudados o comportamento dos gafanhotos-do-deserto. Esse estudo mostrou que a monoamina neurotransmissora conhecida por serotonina é a responsável por fazer com que gafanhotos-do-deserto que vivem de forma solitária passem para uma fase gregária. Fase está de formar nuvens (enxames) de ataques ao alimento.

Para análise algorítmica foi desenvolvido um modelo matemático por base no comportamento do inseto, e vinte e três funções teste foram empregadas, a fim de aferir o desempenho do algoritmo proposto em termos de prospecção, exploração e de convergência. Assim como, um teste computacional para solução do problema do despacho econômico.

O algoritmo do LSO veio de inspiração do comportamento de formação de enxames de gafanhotos-do-deserto, e por estudos feitos na literatura que retratam algoritmos bioinspirados na natureza que emulam comportamento de enxames de espécies, tais como, o algoritmo de colônias de formigas e o algoritmo de otimização por enxames de partículas.

Os resultados mostraram que o *Locust Swarm Optimizer* foi capaz de fornecer resultados competitivos em comparação com populares algoritmos bioinspirados, tais como *Particle Swarm Optimization, Genetic Algorithm*, *Ant Colony Optimization* e *Cuckoo Search*.

Primeiramente, os resultados adquiridos com a aplicação dos algoritmos nas funções *unimodais*, mostraram a exploração superior do algoritmo *Locust Swarm Optimizer* em encontrar a solução ótima para a maioria das funções *benchmarks*.

Em segundo lugar, a capacidade de exploração do *Locust Swarm Optimizer* foi confirmada pelos resultados de funções multimodais e de dimensão fixa.

Em terceiro, a análise de convergência *Locust Swarm Optimizer* confirmou a convergência do algoritmo para a maioria dos resultados.

Finalmente, no problema do despacho econômico, o *Locust Swarm Optimizer* conseguiu encontrar o resultado esperado para o problema a se resolver, uma vez comparado com os resultados das demais técnicas.

Desta forma, pode-se concluir que o *Locust Swarm Optimizer* é um novo método de IA que se trabalha de forma bioinspirada para otimização de problemas complexos e que pode-se ser aplicado a diversas áreas da ciência e tecnologia.

5.2 Trabalhos futuros

Em trabalhos futuros, pretende-se melhorar o algoritmo para que se possa empregá-lo em outras áreas. O *Locust Swarm Optimizer* se assemelha com o *Particle Swarm Optimization* e *Ant Colony Optimization*, uma vez que os gafanhotos têm seu comportamento semelhante a partículas em um espaço. Assim, o LSO poderá ser aplicado e testado, como por exemplo, em problemas que envolvem:

- Evolução de redes neurais artificiais;
- Extração de regras de RNAs;
- Escalonamento de tarefas (*Multi-objective Job Shop Scheduling*);
- Roteamento de veículos (*Capacitated Vehicle Routing*);
- Caminho ótimo para operações de perfuração automatizadas;
- Mineração de dados para tarefas de classificação;
- Posicionamento de bases em computação móvel;
- Aproximação poligonal ótima de curvas digitais.

Referências

[1] A. S. Tanenbaum, D. J. Wetherall, *Computer Networks*, 5th ed. Boston, USA: Prentice Hall, 2011.

[2] M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. Dissertation, Polytechnic di Milano, Italy, 1992.

[3] X.-S. Yang, S. Deb, "Cuckoo search via Lévy flights," in *Proc. of the World Congress on Nature & Biologically Inspired Computing(NaBIC2009)*. Coimbatore, India: IEEE, 2009, pp. 210–214.

[4] J. Kennedy, R. Eberhart, "Particle swarm optimization," in *Proc. of the IEEE International Conference on Neural Networks*. IEEE, 1995, pp. 1942–1948.

[5] W. Hu, G. G. Yen, "Adaptive multiobjective particle swarm optimization based on parallel cell coordinate system," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 1, pp. 1–18, 2015.

[6] Y. Wang, H. X. Li, G. G. Yen, W. Song, "MOMMOP: Multiobjective optimization for locating multiple optimal solutions of multimodal optimization problems," *IEEE Transactions on Cybernetics*, vol. 45, no. 4, pp. 830–843, 2015.

[7] K. Margellos, M. Prandini, J. Lygeros, "On the connection between compression learning and scenario based single-stage and cascading optimization problems," *IEEE Transactions on Automatic Control*, vol. 60, no. 10, pp. 2716–2721, 2015.

[8] A. H. Mogos, B. Mogos, A. M. Florea, "A voting approach for comparing several swarm intelligence algorithms," in *Proc. of the 20th International Conference on Control Systems and Computer Science*. Bucharest, Romania: IEEE, 2015, pp. 287–293.

[9] D. Dibblee, J. Maltese, B. M. Ombuki-Berman, A. P. Engelbrecht, "Vector-evaluated particle swarm optimization with local search," in *Proc. of the IEEE Congress on Evolutionary Computation (CEC)*. Sendai, Japan: IEEE, 2015, pp. 187–195.

[10] F. Liu, L. Lin, L. Jiao, L. Li, S. Yang, B. Hou, H. Ma, L. Yang, J. Xu, "Nonconvex compressed sensing by nature-inspired optimization algorithms," *IEEE Transactions on Cybernetics*, vol. 45, no. 5, pp. 1042–1053, May 2015.

[11] P. K, B. R, "A nature inspired swarm based stellar-mass black hole for engineering optimization," in *Proc. of the IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. Tamil Nadu, India: IEEE, 2015, pp. 1–8.

[12] M. K. Ahirwal, A. Kumar, G. K. Singh, "Sub-band adaptive filtering method for electroencephalography/event related potential signal using nature inspired

optimisation techniques," *IET Science, Measurement Technology*, vol. 9, no. 8, pp. 987–997, 2015.

[13] H. Khosravi, M. Abolfazli-E, M. R. Akbarzadeh-T, "A nature inspired transition from differential evolution to particle swarm optimization," in *Proc. of the 5th International Conference on Computer and Knowledge Engineering (ICCKE)*. IEEE, 2015, pp. 87–92.

[14] R. Babers, A. E. Hassanien, N. I. Ghali, "A nature-inspired metaheuristic lion optimization algorithm for community detection," in *Proc. of the 11th International Computer Engineering Conference (ICENCO)*. Cairo, Egypt: IEEE, 2015, pp. 217–222.

[15] P. Norvig, S. Russell, *Inteligência Artificial*. Rio de Janeiro: Elsevier Brasil, 2014, vol. 1.

[16] J. Haugeland, *Artificial Intelligence: The Very Idea*, MIT Press, 1985.

[17] R. E. Bellman, *An Introduction to Artificial Intelligence: Can Computers Think?* Boyd & Fraser Publishing Company, 1978.

[18] D. McDermott, E. Charniak, *Introduction to Artificial Intelligence*, Addison-Wesley, 1985.

[19] P. H. Winston, *Artificial Intelligence*. Addison-Wesley Pub Co, 1992.

[20] R. Kurzweil, *The Age of Intelligent Machines*, MIT Press Cambridge, 1990.

[21] E. Rich, K. Knight, *Artificial intelligence*. McGraw-Hill, 1991, vol. 199, no. 1.

[22] D. L. Poole, A. K. Mackworth, R. Goebel, *Computational intelligence: a logical approach*. Oxford University Press, 1998, vol. 1.

[23] N. J. Nilsson, *Artificial Intelligence: A New Synthesis*, Morgan Kauffman Publishers, 1998.

[24] S. O. Rezende, *Sistemas Inteligentes: Fundamentos e Aplicações*, Editora Manole Ltda, 2003.

[25] D. Barone, *Sociedades Artificiais: A Nova Fronteira da Inteligência nas Máquinas*. Porto Alegre: Bookman, 2003.

[26] M. X. Py, "Sistemas especialistas: uma introdução," *Universidade Federal do Rio Grande do Sul*, vol. 10, no. 11, pp. 1–10, 2009.

[27] A. A. Turing, "Computing Machinery and Intelligence," *Mind*, vol. 59, pp. 433–450, 1950.

[28] G. F. Luger, *Inteligência Artificial*, São Paulo: Pearson Education do Brasil, 2013.

[29] C. Kotteeswaran, A. Rajesh, "A survey of diverse nature bio-inspired computing models," in *Proc. of 2nd International Conference on Current Trends in Engineering and Technology (ICCTET)*. IEEE, 2014, pp. 120–124.

[30] D. H. Ballard, *An Introduction to Natural Computation*, MIT Press, 1999.

[31] L. N. De Castro, *Fundamentals of natural computing: basic concepts, algorithms, and applications*. CRC Press, 2006.

[32] K. De Jong, "Evolutionary Computation: A Unified Approach," in *Proc. of the 14th annual conference companion on Genetic and evolutionary computation*. Pennsylvania, USA: ACM, 2012, pp. 737–750.

[33] K. A. De Jong, *Evolutionary Computation: A Unified Approach*, London, England: MIT Press, 2007.

[34] A. E. Eiben, J. E. Smith, *Introduction to Evolutionary Computing,* Berlin: Springer, 2003, vol. 53.

[35] P. H. R. Gabriel, A. C. B. Delbem, *Fundamentos de Algoritmos Evolutivos*, USP, 2008.

[36] I. Rechenberg, "Cybernetic solution path of an experimental problem," *Ministry of Aviation, Royal Aircraft Establishment*, 1965.

[37] R. Linden, *Algoritmos Genéticos - Uma Importante Ferramenta da Inteligência Computacional*, Rio de Janeiro: Editora Brasport, 2006.

[38] I. Rechenberg, *Evolutionsstrategie Optimierung technischer Systemenach Prinzipien der biologishen Evolution*. Germany: Frommann-Holzboog Verlag, 1973.

[39] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. Inglaterra: U Michigan Press, 1975.

[40] C. Yang, D. Simon, "A new particle swarm optimization technique," in *Proc. of the 18th International Conference on Systems Engineering (ICSEng'05)*. IEEE, 2005, pp. 164–169.

[41] B. Soudan, M. Saad, "An evolutionary dynamic population size pso implementation," in *Proc. of the 3rd International Conference on Information and Communication Technologies: From Theory to Applications (ICTTA 2008)*. IEEE, 2008, pp. 1–5.

[42] Y. Shi, R. Eberhart, "A modified particle swarm optimizer," in *Proc. of the IEEE International Conference on Evolutionary Computation Proceedings*. IEEE World Congress on Computational Intelligence, 1998, pp. 69–73.

[43] J. Xin, G. Chen, Y. Hai, "A particle swarm optimizer with multistage linearly-decreasing inertia weight," in *Proc. of the International Joint Conference on Computational Sciences and Optimization. IEEE*, 2009, pp. 505–508.

[44] L. E. Da, Silva, "Sistema híbrido metaheurístico baseado em colônia de formigas paraconsistentes aplicado a problemas de otimização em redes inteligentes," Tese de doutorado, Universidade Federal de Itajubá, Itajubá, 2012.

[45] R. F. Tavares Neto, M. Godinho Filho, “Otimização por colônia de formigas para o problema de sequenciamento de tarefas em uma única máquina com terceirização permitida,” *Gestão e Produção*, vol. 20, no. 1, pp. 76–86, 2013.

[46] K. Socha, M. Dorigo, “Ant colony optimization for continuous domains,” *European Journal of Operational Research*, vol. 185, no. 3, pp. 1155–1173, 2008.

[47] R. B. Payne, M. D. Sorensen, *The cuckoos*. New York, USA: Oxford University Press, 2005, vol. 15.

[48] C. T. Brown, L. S. Liebovitch, R. Glendon, “Lévy flights in dobe ju/’hoansi foraging patterns,” *Human Ecology*, vol. 35, no. 1, pp. 129–138, 2007.

[49] A. H. Gandomi, X.-S. Yang, A. H. Alavi, “Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems,” *Engineering with Computers*, vol. 29, no. 1, pp. 17–35, 2013.

[50] L. N. de Castro, J. Timmis, “Artificial immune systems: a novel paradigm to pattern recognition,” *Artificial Neural Networks in Pattern Recognition*, vol. 1, pp. 67–84, 2002.

[51] L. N. de Castro, “Engenharia imunológica: Desenvolvimento e aplicação de ferramentas computacionais inspiradas em sistemas imunológicos artificiais,” Tese de doutorado, Universidade Estadual de Campinas, Campinas, São Paulo, 2001.

[52] S. S. Souza, R. Romero, J. Pereira, J. T. Saraiva, “Artificial immune algorithm applied to distribution system reconfiguration with variable demand,” *International Journal of Electrical Power & Energy Systems*, vol. 82, pp. 561–568, 2016.

[53] N. S. Halvaiee, M. K. Akbari, “A novel model for credit card fraud detection using artificial immune systems,” *Applied Soft Computing*, vol. 24, pp. 40–49, 2014.

[54] M.-H. Chen, P.-C. Chang, J.-L. Wu, “A population-based incremental learning approach with artificial immune system for network intrusion detection,” *Engineering Applications of Artificial Intelligence*, vol. 51, pp. 171–181, 2016.

[55] H. S. Bernardino, H. J. C. Barbosa, *Artificial Immune Systems for Optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[56] L. N. De, Castro, F. J. Von, Zuben, “The clonal selection algorithm with engineering applications,” in *Proc. of the Genetic and Evolutionary Computation Conference (GECCO)*. GECCO, 2000, pp. 36–39.

[57] ——, “Learning and optimization using the clonal selection principle,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 239–251, 2002.

[58] T. M. Ribeiro, N. R. P. da Cunha, F. d. F. de Lima Sousa, “Method for computational intelligence based on behavior of grasshoppers,” in *Proc. of the Zone I Conference on American Society for Engineering Education (ASEE2014)*. ASEE2014, 2014, pp. 1–4.

[59] T. M. Ribeiro, V. L. Paucar, M. R. P. dos Santos, N. R. P. da Cunha, “Proposta para detecção de anomalias de dados na nuvem baseado no comportamento de enxame

de gafanhotos," in *Proc. of the XXI Congresso Brasileiro de Automática (CBA2016)*, 2016, pp. 1–6.

[60] M. L. Anstey, S. M. Rogers, S. R. Ott, M. Burrows, S. J. Simpson, "Serotonin Mediates Behavioral Gregarization Underlying Swarm Formation in Desert Locusts," *Science*, vol. 323, no. 5914, pp. 627–630, Jan. 2009.

[61] S. M. Rogers, D. A. Cullen, M. L. Anstey, M. Burrows, E. Despland, T. Dodgson, T. Matheson, S. R. Ott, K. Stettin, G. A. Sword, "Rapid behavioural gregarization in the desert locust, schistocerca gregaria entails synchronous changes in both activity and attraction to conspecifics," *Journal of Insect Physiology*, vol. 65, pp. 9–26, June 2014.

[62] A. B. H. Ferreira, *Novo Dicionário da Língua Portuguesa*. Rio de Janeiro: Nova Fronteira, 1986.

[63] E. Costa, Neto, A. Duarte, A. Jesus, A. Ribeiro, F. Lima, M. Nascimento, N. Reis, T. Henrique, W. Jesus, "Antropoentomofagia-insetos na alimentação humana," *Antropoentomofagia-Sobre o Consumo de Insetos*, vol. 1, pp. 17–37, 2011.

[64] H. Song, "Density-dependent phase polyphenism in nonmodel locusts: A minireview," *Psyche: A Journal of Entomology*, vol. 2011, pp. 1–16, 2010.

[65] C. Hotta, "Gafanhotos em polvorosa," 2016, URL: http://goo.gl/9m87Xl [Acesso em: 02 maio 2016].

[66] E. Yong, "The swarm-maker molecule: how serotonin transforms solitary locusts into social ones," 2009, URL: http://goo.gl/TGoP1i [Acesso em: 02 maio 2016].

[67] D. Treinomestre.com.br, "Serotonina: o que, sintomas da falta dessa substância e como estimular sua produção," 2013, URL: http://goo.gl/jR9qY6 [Acesso em: 02 maio 2016].

[68] M. A. Medeiros, "As pragas também morrem pela boca," 2000, URL: http://goo.gl/gWvXcZ [Acesso em: 03 maio 2016].

[69] Y. Xin, Y. Liu, G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary computation*, vol. 3, no. 2, pp. 82–102, 1999.

[70] J. G. Digalakis, K. G. Margaritis, "An experimental study of benchmarking functions for genetic algorithms," *IEEE International Conference on Systems*, Man, and Cybernetics, vol. 5, pp. 3810–3815, 2000.

[71] M. Jamil, X.-S. Yang, "A literature survey of benchmark functions for global optimization problems," *International Journal of Mathematical Modelling and Numerical Optimization*, vol. 4, no. 2, pp. 150–194, 2013.

[72] E. Cuevas, M. Cienfuegos, D. Zaldívar, M. Pérez-Cisneros, "A swarm optimization algorithm inspired in the behavior of the social spider," *Expert Systems with Applications*, vol. 40, no. 16, pp. 6374–6384, 2013.

[73] S. Mirjalili, A. Lewis, “S-shaped versus v-shaped transfer functions for binary particle swarm optimization,” *Swarm and Evolutionary Computation*, vol. 9, pp. 1–14, 2013.

[74] S. Mirjalili, S. M. Mirjalili, X.-S. Yang, “Binary bat algorithm,” *Neural Computing and Applications*, vol. 25, no. 3-4, pp. 663–681, 2014.

[75] S. Mirjalili, S. M. Mirjalili, A. Lewis, “Grey wolf optimizer,” *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[76] K. Socha, M. Dorigo, “Ant colony optimization for continuous domains,” *European journal of Operational Research*, vol. 185, no. 3, pp. 1155–1173, 2008.

[77] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, 2nd ed., United Kingdom: Luniver Press, 2010.

[78] D. H. Wolpert, W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67-82, 1997.

[79] A. J. Wood, B. F. Wollenberg, *Power Generation, Operation, and Control*, 2nd ed., John Wiley & Sons, 1996.

[80] A. S. Tanenbaum, *Modern Operating Systems*, Pearson Education, 2009.

[81] S. Srinoy, W. Kurutach, “Combination artificial ant clustering and k-pso clustering approach to network security model,” in *Proc. of the International Conference on Hybrid Information Technology*. IEEE, 2006, pp. 128–134.

[82] A. H. Hamamoto, L. F. Carvalho, M. L. Proenca, “ACO and GA metaheuristics for anomaly detection,” in *Proc. of the 34th International Conference of the Chilean Computer Science Society (SCCC)*. IEEE, 2015, pp. 1–6.

[83] Z. Luo, R. Wan, X. Si, “An improved aco-based security routing protocol for wireless sensor networks,” in *Proc. of the International Conference on Computer Sciences and Applications (CSA)*. IEEE, 2013, pp. 90–93.

[84] A. Erramilli, O. Narayan, W. Willinger, “Experimental queueing analysis with long-range dependent packet traffic,” *IEEE/ACM Transactions on Networkingh*, vol. 4, no. 2, pp. 209–223, 1996.

[85] D. L. Applegate, R. E. Bixby, V. Chvatal, W. J. Cook, *The Traveling Salesman Problem: A Computational Study*, Princeton University Press, 2011.

[86] G. R. Ash, *Dynamic Routing in Telecommunications Networks*, McGraw-Hill Professional, 1997.