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

ANÁLISE DO DESEMPENHO DE MÉTODOS DE INTELIGÊNCIA ARTIFICIAL BASEADOS NO COMPORTAMENTO DAS PLANTAS

MARÍLIA MARTA GOMES ORQUIZA DE AZEVEDO

São Luís – MA, Brasil
Fevereiro, 2017

MARÍLIA MARTA GOMES ORQUIZA DE AZEVEDO

ANÁLISE DO DESEMPENHO DE MÉTODOS DE INTELIGÊNCIA ARTIFICIAL BASEADOS NO COMPORTAMENTO DAS PLANTAS

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

Orientador: Prof. Dr. Vicente Leonardo Paucar Casas

São Luís – MA, Brasil
Fevereiro, 2017

Gomes Orquiza de Azevedo, Marília Marta.
ANÁLISE DO DESEMPENHO DE MÉTODOS DE INTELIGÊNCIA ARTIFICIAL BASEADOS NO COMPORTAMENTO DAS PLANTAS / Marília Marta Gomes Orquiza de Azevedo. - 2017.
80 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. Algoritmo da colonização das ervas daninhas. 2. Algoritmo de polinização das flores. 3. Algoritmo de propagação do morango. 4. Inteligência artificial. 5. Inteligência das plantas. I. Paucar Casas, Vicente Leonardo. II. Titulo.

ANÁLISE DO DESEMPENHO DE MÉTODOS DE INTELIGÊNCIA ARTIFICIAL BASEADOS NO COMPORTAMENTO DAS PLANTAS

MARÍLIA MARTA GOMES ORQUIZA DE AZEVEDO

Dissertação de Mestrado 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 Cícero Pavão Lopes
UFMA
(Membro da Banca Examinadora)

Este Trabalho é dedicado aos meus pais: Cândido (in memoriam) e Iolete. A minha avó Delfina (in memoriam). Ao meu esposo Rafael e minha irmã Flávia.

Agradecimentos

A Deus pelo amor incondicional, pelos livramentos e bênçãos recebidas, pelo cuidado e misericórdia, e por nunca me deixar só.

Aos meus pais, Cândido e Iolete, por todo o sacrifício feito pela minha educação e bem estar. Pelos ensinamentos, companheirismo, e ajuda. Pela presença em minha vida e principalmente pelo amor sem medidas.

Ao meu esposo Rafael por toda a paciência, ajuda, companheirismo nesses dois anos de mestrado. Pelo amor, carinho, cuidado e incentivo, por muitas vezes enxugar minhas lágrimas e fazer perceber que eu conseguiria.

À minha irmã Flávia que sempre com amor e paciência me atura e faz enxergar o lado bom das coisas.

Ao professor Dr. Vicente Leonardo Paucar pela oportunidade dada, pela orientação, preocupação, paciência, dedicação e apoio.

Aos alunos do Laboratório de Mercados Elétricos pelo grande apoio, por toda ajuda dada e pela amizade.

A CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pelo apoio financeiro.

"A menos que modifiquemos a nossa maneira de pensar, não seremos capazes de resolver os problemas causados pela forma como nos acostumamos a ver o mundo".

(Albert Einstein)

Resumo

A inteligência artificial (IA) é um ramo da ciência da computação que estuda o comportamento inteligente dos seres vivos e imita essa inteligência implantando-a em programas de computador, máquinas e sistemas para resolver problemas relacionados à busca, otimização, planejamento, controle, automação, etc. Uma das áreas da inteligência artificial é a computação evolutiva, que é inspirada pelo princípio da evolução natural das espécies. Dentro da computação evolutiva vários métodos baseados na informação de plantas têm sido recentemente proposto. Como as plantas sobrevivem e se adaptam em ambientes agressivos tem despertado grande interesse dos pesquisadores em IA. O ciclo de vida de uma planta é extremamente intrigante. A maneira como as plantas se reproduzem, propagam, dispersam suas sementes e selecionam as mais resistentes é, sem dúvida, uma evidência de inteligência das plantas quando otimizam sua existência. Nesse sentido, diversos algoritmos computacionais baseados no ciclo de vida inteligente das plantas têm sido propostos nos anos recentes, esses algoritmos são, em muitos casos, simples de implementar e muito eficientes na solução de problemas complexos. Neste trabalho é analisado o desempenho de alguns desses algoritmos, o algoritmo de polinização de flores, o algoritmo de planta de morango, otimização invasiva de ervas daninhas e algoritmo do ciclo de vida da planta, todos baseados no comportamento inteligente das plantas, quando aplicados à otimização de funções teste e também comparados com algoritmos genéticos clássicos.

Palavras-chave: Inteligência artificial, inteligência das plantas, algoritmo de polinização das flores, algoritmo de propagação do morango, algoritmo da colonização das ervas daninhas.

Abstract

Artificial intelligence (AI) is a branch of computer science that studies the intelligent behavior of living beings, and mimics this intelligence by deploying it in computer programs, machines and systems in order to solve problems related to searching, optimization, planning, control, automation, etc. One of the areas of artificial intelligence is evolutionary computation, which is inspired by the principle of natural evolution of species. Within the evolutionary computation several methods based on the intelligence of plants have been recently proposed. How the plants survive and adapt in harsh environments has aroused great interest of researchers in AI. It is remarkable that the life cycle of a plant is extremely intriguing. The way the plants reproduce, propagate, disperse their seeds and select the most resistant is undoubtedly an evidence of intelligence of plants when optimize their existence. In this sense, several computer algorithms based on the intelligent lifecycle of plants have been proposed recently, these algorithms are in many cases, simple to implement, and very efficient in solving complex problems. In this work, the performance of some algorithms, the flower pollination algorithm, strawberry plant algorithm, invasive weed optimization and plant life cycle algorithm, all of them based on the intelligent behavior of plants, are analyzed when applied to optimization of test functions, and they are also compared with classical genetic algorithms.

Keywords: Artificial intelligence, intelligence of plants, flower pollination algorithm, strawberry propagation algorithm, weed colonization algorithm.

Lista de Figuras

Página

Lista de Tabelas

Página

Lista de Abreviaturas

IA	: Inteligência Artificial
AG	: Algoritmos Genéticos
PSO	: *Particle Swarm Optimization*
IWO	: *Invasive Weed Optimization*
SBA	: *Strawberry Algorithm*
IE	: Inteligência de Enxames
ACO	: *Ant Colony Optimization*
FPA	: *Flower Pollination Algorithm*
PLC	: *Plant Life Cycle*
IWO-M	: *Invasive Weed Optimization Modified*

Lista de Símbolos

b_i	: Solução
g_*	: Pólen mais apto
L	: Tamanho do passo
t	: Iteração atual
s	: Etapas do intervalo
s_0	:Etapa inicial do intervalo
Γ	: Função gama padrão
N_iter	: Número máximo de iterações
tol	: Tolerância para achar a solução
f_{min}	: Melhor solução encontrada
d	: Espaço de busca dimensional
σ	: Desvio padrão
$\sigma_{inicial}$	: Desvio padrão inicial
σ_{final}	: Desvio padrão final
n	: Índice de modulação linear
N_{min}	: Número mínimo de plantas
N	: Número de plantas
L_{root}	: Distância entre as raízes
L_{runner}	: Distância entre os corredores
fit	: Função *fitness*
r_1	: Matriz aleatória
r_2	: Matriz aleatória
$X(t)$	: Planta mãe
X_{prop}	: Planta filha
P_{dis}	: Distância entre plantas
P_c	: Qualidade da planta
P_n	: Probabilidade aleatória
P_s	: Número médio de sementes

Sumário

Página

Capítulo 1

1. Introdução

A Inteligência Artificial (IA) é uma das mais recentes ciências, estuda o comportamento inteligente dos seres vivos e imita essa inteligência implantando-a em programas de computador, máquinas e sistemas para resolver problemas relacionados à busca, planejamento, controle, automação, etc. [1]. Nas últimas décadas os estudos na área de IA têm crescido bastante, e essa ciência deixou de imitar somente o comportamento humano, imitando também o comportamento inteligente de animais e vegetais.

A natureza é um grande exemplo de resolução de problemas complexos. Tanto o reino animal quanto o reino vegetal sobrevive, reproduz e evolui de forma intrigante, surpreendente e encantadora. Esse comportamento inteligente dos seres da natureza têm despertado grande interesse de estudiosos de IA, os inspirando à elaboração de algoritmos baseados em tais comportamentos, algoritmos capazes de resolver problemas complexos, a exemplo os de otimização, computação e engenharia, é a então chamada inteligência artificial bioinspirada.

Os algoritmos bioinspirados são estatísticos, dependentes de dados de entrada, que serão combinados aleatoriamente, dessa combinação sairão resultados que serão guardados e utilizados para novas combinações até que alguma condição de parada seja atendida e o real resultado esperado seja encontrado, resultado este que satisfará as características da otimização. Há diversos algoritmos bioinspirados, os considerados clássicos e também evolutivos, são: Algoritmos Genéticos (*Genetic Algorithm* - GA) e Otimização de Enxames de Partículas (*Particle Swarm Optimization* - PSO). Os mais recentes algoritmos probabilísticos são baseados na inteligência das plantas, como o de Otimização da Erva Daninha Invasora (*Invasive Weed Optimization* - IWO) e Algoritmo da Planta do Morango (*Strawberry Algorithm* - SBA).

1.1. Formulação do problema

O crescente estudo na área de inteligência artificial bioinspirada tem disponibilizado inúmeros algoritmos computacionais que se julgam capazes de resolver os diversos problemas a que são empregados. Todos esses algoritmos têm suas vantagens e desvantagens, e suas características e limitações os diferem e os tornam a melhor solução.

A todo momento algoritmos inspirados na inteligência dos vegetais surgem, nem sempre novos, mas adaptações dos já existentes (ou do tipo híbrido). Já não se sabe qual algoritmo é o mais recente, o mais inovador ou eficiente. Surgem assim as dúvidas sobre: "Quais os algoritmos bioinspirados existentes?", "Qual o algoritmo mais eficiente?", "Em que esse algoritmo se destaca?"

1.2. Objetivos

O objetivo geral é:

- Contribuir com o estudo de métodos de inteligência artificial inspirados no comportamento inteligente no reino vegetal, através de uma comparação de desempenho entre alguns algoritmos evolutivos baseados no comportamento inteligente de plantas.

Os objetivos específicos são:

- Analisar o desempenho de métodos de inteligência artificial baseados no comportamento de plantas, sementes e flores;
- Aplicar os métodos de inteligência artificial bioinspirados para resolver funções de otimização *benchmarking* uni-modais e multimodais;
- Comparar o desempenho de métodos de inteligência artificial bioinspirados com outros métodos de inteligência artificial clássicos da literatura.

1.3. Justificativa

O estudo, o desenvolvimento e a aplicação dos algoritmos inspirados no reino vegetal têm recebido bastante atenção na última década. A literatura existente revela que, a cada ano, vários algoritmos inspirados no comportamento inteligente dos animais e vegetais são desenvolvidos. Porém, uma grande atenção é dada aos clássicos algoritmos bioinspirados, tais como: algoritmos genéticos, enxame de partículas, evolução diferencial e colônia de formigas. Já os algoritmos bioinspirados no reino vegetal, poucas referências são encontradas na literatura. Portanto, este trabalho pretende contribuir com um estudo aprofundado sobre os algoritmos bioinspirados no reino vegetal com o intuito de analisar seu desempenho perante outros algoritmos clássicos bioinspirados.

1.4. Metodologia

Para o desenvolvimento da presente pesquisa proposta foi adotada a metodologia da pesquisa científica.

A princípio foi realizada uma pesquisa bibliográfica para levantar o estado da arte acerca das técnicas bioinspiradas.

Em seguida foram estudados os algoritmos com inspiração biológica da inteligência natural das plantas. Paralelamente, cada algoritmo bioinspirado baseado foi implementado em seu pseudocódigo.

Na sequência, diversas funções *benchmarking* uni-modais e multimodais foram selecionadas com o objetivo de aplicá-las aos algoritmos bioinspirados implementados na etapa anterior. Igualmente, todos os algoritmos com inspiração biológica da inteligência natural das plantas serão comparados com os clássicos algoritmos bioinspirados.

A verificação do andamento do desenvolvimento deste trabalho foi realizada através de seminários trimestrais de acompanhamento baseado no regime e normas do Programa de Pós-graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão (UFMA).

1.5. Estrutura do trabalho

Este trabalho contém seis capítulos organizados da seguinte maneira.

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

O Capítulo 2 apresenta conceitos relativos aos métodos de inteligência artificial bioinspirados.

O Capítulo 3 apresenta os métodos de inteligência artificial inspirados no comportamento inteligente das plantas.

O Capítulo 4 apresenta as performances e as comparações dos algoritmos bioinspirados na inteligência das plantas, utilizando para testes desses algoritmos funções de otimização *benchmarking*.

O Capítulo 5 apresenta uma modificação do algoritmo IWO.

O Capítulo 6 contém as conclusões deste trabalho.

Capítulo 2

2. Métodos de inteligência artificial bioinspirados

Esse capítulo é inicializado com uma discussão sobre os conceitos essenciais para o entendimento dos próximos capítulos. A seção 2.1 é uma introdução ao capítulo. Na seção 2.2 são apresentados conceitos básicos sobre inteligência artificial, com sua classificação e aplicações. A seção 2.3 expõe conceitos de computação evolutiva e alguns algoritmos bioinspirados clássicos.

2.1. Introdução

Em informática, a inteligência artificial (IA) está relacionada com o desenvolvimento de sistemas informáticos inteligentes, incluindo máquinas e métodos. Há diversos estudos na área de inteligência artificial, mas ainda não houve um acordo sobre uma única definição de IA. A computação evolutiva (EC) é uma importante área recente da IA, que está interessada no estudo da inteligência dos seres vivos e sua implementação em algoritmos e sistemas computacionais. Tais métodos de EC baseiam-se em como animais e plantas sobrevivem e evoluem em habitats nem sempre favoráveis a si mesmos, demonstrando uma intrigante e surpreendentemente inteligência. Aplicação de algoritmos EC é adequado para diferentes áreas da ciência e engenharia [1].

Neste capítulo serão abordados conceitos de inteligência artificial, computação evolutiva e alguns métodos inspirados no comportamento de animais.

2.2. Inteligência artificial

Inteligência Artificial (IA), nomenclatura dada a um campo de pesquisa científica em um seminário em Dartmouth em 1956, ainda é difícil de ser definida [1]. A discordância entre seus pesquisadores em relação a uma só definição não chegou ao fim. Em diversos livros e estudos sobre esta ciência cada um dos autores a define de acordo com sua área ou subcampo de pesquisa.

Russel [1] assegura que a IA é definida baseada em duas dimensões: pensamento e comportamento; e em duas entidades: humana e racional. Como segue:

- Sistemas que pensam como os humanos;
- Sistemas que pensam racionalmente;
- Sistemas que atuam como seres humanos;
- Sistemas que atuam racionalmente.

Acredita-se que todos os quatro grupos de pesquisa estão sendo, ou foram, estudados, em diversos subcampos.

Para Luger [2], a inteligência artificial é um ramo da ciência da computação que se ocupa da automação do comportamento inteligente. Esta definição é a mais apropriada para este estudo.

A inteligência artificial é uma ciência relativamente jovem, pois os primeiros trabalhos na área surgiram logo após a Segunda Guerra Mundial. Estuda o comportamento inteligente dos seres vivos em geral e o automatiza. O campo da IA tenta não apenas compreender, mas também construir entidades inteligentes [1].

A fundamentação teórica da IA vem de diversas ciências como: Filosofia, com os estudos sobre dualismo, empirismo, positivismo lógico, por exemplo; Matemática, com os algoritmos, probabilidade, intratabilidade; Economia, com teoria dos jogos, teoria da decisão; Neurociência com o estudo do funcionamento do cérebro; a Psicologia, com a ciência cognitiva; entre outras.

Seguindo essa ideia, os problemas de pesquisas da IA incluem: planejamento, aprendizagem, conhecimento, raciocínio, percepção, entre outros. A IA agrega diversas áreas das ciências e profissões, como: ciência da computação, matemática, filosofia, neurociência.

O campo da inteligência artificial é muito amplo, mas pode-se dizer que se divide em duas áreas principais, a IA clássica e a IA moderna. Tem como principal preocupação dos pesquisadores a busca e a representação do conhecimento.

A IA clássica abrange as subáreas de resolução de problemas, sistemas baseados no conhecimento, sistemas especialistas, etc. Já a IA moderna tem como subáreas estudos em redes neurais, logica *fuzzy*, aprendizado de máquina, agentes inteligentes e computação evolutiva, computação inteligente, por exemplo.

2.3. Computação evolutiva

A evolução pode ser definida como um processo de otimização, que tem como objetivo melhorar a capacidade de um organismo (sistema) para sobreviver em ambientes competitivos e/ou em constante mudanças. A evolução biológica baseada na teoria da seleção natural de Charles Darwin pode ser resumida como: em um mundo com recursos limitados e populações estáveis, cada indivíduo compete com os outros por sua sobrevivência. Os indivíduos com as "melhores" características (traços) são mais propensos a sobreviver e reproduzir, e essas características serão transmitidas aos seus descendentes. Estas características desejáveis são herdadas pelas seguintes gerações, e (ao longo do tempo) tornam-se dominantes entre a população [3].

Computação evolutiva, área de pesquisa da inteligência artificial, é um paradigma de computação baseada na teoria darwiniana da evolução através da seleção natural. Para os pesquisadores de computação evolutiva, os mecanismos da evolução são inspiradores e adequados para solução de alguns problemas computacionais, como: a busca através de um grande número de possibilidades de soluções, além da implementação de sistemas computacionais capazes de computar dados complexos que são difíceis de programar manualmente [4].

Na computação evolutiva, as regras são tipicamente "seleção natural", com variação devido ao crossover e/ou mutação; O esperado comportamento emergente é o *design* de soluções de alta qualidade para problemas difíceis e a capacidade de adaptar estas soluções face a um ambiente em mudança [4]. Os algoritmos baseados neste paradigma possuem características como auto-organização e comportamento adaptativo, são estocásticos, utilizam a técnica de tentativa e erro para encontrar a otimização global. São iterativos, geram uma população inicial de forma aleatória e utilizam um processamento paralelo para atingir o fim desejado. A computação evolutiva tem sido utilizada com sucesso em aplicações, a exemplo: mineração de dados, otimização combinatória, diagnóstico de falhas, classificação, agrupamento, programação e aproximação de séries temporais [3].

2.3.1. Algoritmos genéticos

Algoritmos genéticos (AG) são uma das principais estratégias de busca e otimização da computação evolutiva, inspirado no princípio darwiniano de seleção natural

e reprodução genética. Foram introduzidos por John Holland em 1975 e popularizados por um dos seus alunos, David Goldberg, 1989. Tem como características a busca estocástica, paralelismo, generalidades e facilidade no uso de restrições [5].

Os algoritmos genéticos trabalham com uma população de soluções para um dado problema em vez de tratar cada solução individualmente. Esse conjunto de soluções passa por uma série de seleções, pareamentos e recombinações com o intuito de gerar novas soluções e de melhorar as aptidões dos indivíduos.

Os principais operadores de condução de um AG é a seleção (para modelar a sobrevivência do mais apto) e recombinação através da aplicação de um operador de crossover (para a reprodução do modelo) [3].

Os diversos estudos dos algoritmos genéticos possibilitou inúmeras maneiras de implementação de tal algoritmo, mas todos seguindo os passos do AG Clássico. O primeiro passo é a geração aleatória de uma população inicial de cromossomos, durante o processo evolutivo, esta população é avaliada por uma função *fitness* e cada cromossomo recebe uma aptidão. O segundo passo é a seleção, em que serão selecionados os cromossomos mais aptos, os demais serão descartados. Os cromossomos selecionados podem sofrer recombinações em suas características fundamentais através dos operadores de crossover e mutação, gerando descendentes para a próxima população, garantindo os passos 3 e 4. Estes passos são repetidos até que uma solução satisfatória seja encontrada.

A forma genérica de um algoritmo genético é representada no pseudocódigo da Figura 2.1. Nesse pseudocódigo, o primeiro procedimento consiste na criação de uma população inicial de indivíduos $Pop(x=1)$, em que x representa o contador de gerações. Cada indivíduo de $Pop(x)$ é uma solução candidata do problema. Logo, tem-se uma função de avaliação (*fitness*) ou aptidão da população $Pop(x)$. A continuação se tem um ciclo enquanto ... faça (*while ... do*) em que nessa população de indivíduos será aplicada a operação de cruzamento (*crossover*), mutação e a função de avaliação para verificar se o critério de parada foi alcançado. O critério de parada pode ser, por exemplo, se o mínimo de uma função que a priori sabe-se deve ser zero foi alcançado para uma tolerância pré-especificada.

Algoritmo genético
Seja $Pop(x)$ a população de cromossomos da geração x $x \leftarrow 0$ Inicializar $Pop(x)$ Avaliar $Pop(x)$ **Enquanto** o critério de parada não for satisfeito **faça** $x \leftarrow x + 1$ Selecionar $Pop(x)$ a partir de $Pop(x-1)$ Aplicar crossover sobre $Pop(x)$ Aplicar mutação sobre $Pop(x)$ Avaliar $Pop(x)$ **Fim Enquanto**

Figura 2.1 Pseudocódigo do algoritmo genético. [5]

2.4. Inteligência de enxames

Uma outra classe de algoritmos baseados na natureza é a classe dos algoritmos que se inspiram na Inteligência de Enxame (IE). A IE se originou do estudo de colônias ou enxames de organismos sociais. O comportamento social de organismos (indivíduos) em enxames levou ao *design* de muitos eficientes algoritmos de otimização e agrupamento. Por exemplo, os estudos da simulação graciosa, mas imprevisível, coreografia dos revoada de aves, levou ao algoritmo de otimização de enxames de partículas e estudos do comportamento das formigas resultou em algoritmos de otimização de colônias de formigas [3].

A inteligência de enxames, também referenciada como inteligência de colônias ou inteligência coletiva, é um conjunto de técnicas baseadas no comportamento coletivo de sistemas auto organizados, distribuídos, autônomos, flexíveis e dinâmicos, inspirados nas formigas, abelhas, aranhas, peixes, etc. Os estudos sobre o comportamento complexo dos enxames resultou em sistemas formados por uma população de agentes computacionais simples que possuem a capacidade de perceber e modificar o seu ambiente de maneira local. Esta capacidade torna possível a comunicação entre os agentes, que captam as mudanças no ambiente geradas pelo comportamento de seus congêneres. Embora não

exista uma estrutura centralizada de controle que estabelece como os agentes devem se comportar, e mesmo não havendo um modelo explícito do ambiente, as interações locais entre os agentes geralmente levam ao surgimento de um comportamento global que se aproxima da solução do problema [6].

2.4.1. Colônia de formigas

O comportamento de diferentes espécies tem sido modelado pela abordagem de IE, a exemplo a colônia de formigas. A otimização por colônia de formigas (*Ant Colony Optimization* - ACO) é uma técnica de IA que tem por inspiração o comportamento da colônia de formigas e sua capacidade de encontrar o caminho mais curto, partindo de uma fonte de alimentos até seu ninho, usando o feromônio deixado no caminho [7]. O método foi proposto por Marco Dorigo e Luca Gambardella [7].

O feromônio é uma substância química que permite que seres da mesma espécie se reconheçam e interajam entre si. Essa substância exerce um papel importante na troca de informações entre as formigas para a realização de suas atividades básicas. Existem diferentes tipos de feromônio, sendo utilizados para diferentes fins. O tipo de feromônio também varia de acordo com a espécie e colônia.

Durante seu movimento, as formigas depositam trilhas de feromônio no trecho do solo percorrido. Como as formigas são atraídas pelo feromônio, outras formigas tendem a seguir o mesmo caminho. Inicialmente, formigas se movem de forma aleatória. Ao encontrar um caminho com feromônio, a formiga precisa decidir se segue ou não o caminho encontrado. Se ela seguir o caminho, ela depositará uma quantidade adicional de feromônio, aumentando a chance de que outras formigas percorram este caminho. Caminhos com uma maior quantidade de feromônio atraem uma maior quantidade de formigas [8].

Estudos de colônias de formigas têm contribuído em abundância para o conjunto de algoritmos inteligentes. A modelagem do depósito de feromônio pelas formigas na sua busca por caminhos para fontes de alimento resultou no desenvolvimento de algoritmos de otimização de caminho mais curto. Outras aplicações da otimização de colônias de formigas incluem otimização de roteamento em redes de telecomunicações, coloração gráfica, agendamento e resolução de problema de atribuição [3].

O algoritmo básico que representa o comportamento das formigas está demostrado na Figura 2.2.

Algoritmo colônias de formigas
Inicialize **Repita** *Neste nível, cada execução é chamada* ***iteração*** **Repita** *Neste nível, cada execução é chamada* ***passo*** Cada formiga aplica uma regra de transição para construir a próxima etapa da solução. Aplica-se a atualização local de feromônios. **Até que** *todas as formigas tenham criado uma solução completa* Aplica-se o procedimento de busca local. Aplica-se o procedimento de atualização global de feromônios. **Até que** *o critério de parada seja satisfeito*

Figura 2.2 Pseudocódigo do Algoritmo Colônias de Formigas. [8]

2.4.2. Otimização por enxame de partículas

Outro exemplo de algoritmos baseados na IE é otimização por enxame de partículas (*Particle Swarm Optimization* - PSO). O PSO é um algoritmo de otimização global para lidar com problemas em que uma melhor solução pode ser representada como um ponto ou de superfície num espaço n-dimensional. Foi proposto inicialmente por Kennedy e Eberhart [9], destinado primeiramente para simular o comportamento social, como uma representação estilizada do movimento de organismos em coletivo de pássaros e peixes.

PSO é um procedimento de pesquisa de base populacional, onde os indivíduos, referidos como partículas, são agrupados em um enxame. Cada partícula do enxame representa uma solução candidato para o problema de otimização. Em um sistema de PSO, cada partícula "viaja" através do espaço de busca multidimensional, ajustando sua posição no espaço de busca de acordo com a sua própria experiência e na de partículas vizinhas. Portanto uma partícula faz uso da melhor posição encontrado por si só e a melhor posição dos seus vizinhos a posicionar-se em direção a uma solução ótima. O efeito é que as

partículas “voam” em direção a um ideal, enquanto ainda está procurando uma grande área ao redor a melhor solução atual. O desempenho de cada partícula (isto é, a “proximidade” de uma partícula para o mínimo global) é medido de acordo com uma função de aptidão pré-definida que está relacionada com o problema a ser resolvido [3].

Há diversas variações de PSO, em formas híbridas ou melhoradas, aumentando sua precisão na resolução de problemas complexos, otimizando os resultados. Na Figura 2.3, há uma representação do algoritmo PSO básico, em que P é uma população de partículas e p é uma partícula.

Algoritmo otimização por enxame de partículas

1. Determine o número de partículas p da população.
2. Inicialize aleatoriamente a posição inicial (x) de cada partícula p de P.
3. Atribua uma velocidade inicial (v) igual para todas as partículas.
4. Para cada partícula p em P faça:
    - Calcule sua aptidão $fp = f\ (p)$.
    - Calcule e melhor posição da partícula p até o momento (pB).
5. Descubra a partícula com a melhor aptidão de toda a população (gB).
6. Para cada partícula p em P faça:
    - Atualize a velocidade da partícula.
    - Atualize a posição da partícula.
7. Se condição de término não for alcançada, retorne ao passo 4.

Figura 2.3 Algoritmo de otimização por enxame de partículas. [6]

A natureza está repleta de exemplos de soluções elegantes e eficientes para um grande número de problemas. Estas soluções estão presentes nas mais diversas espécies de seres vivos. É fácil vê-las quando abelhas se organizam para construir uma colmeia, facilitando a sobrevivência do enxame ou espalham o pólen de uma planta, permitindo a fecundação de uma outra planta, entre vários outros exemplos. Estas soluções são, em geral, o resultado de um processo de evolução. Esta á uma das principais características presentes em sistemas biológicos [8]. No capítulo seguinte serão apresentados alguns algoritmos baseados no comportamento inteligente das plantas para resolver problemas evolutivos.

Capítulo 3

3. Métodos de inteligência artificial baseados na inteligência das plantas

Neste capítulo, alguns métodos de inteligência artificial inspirados na inteligência das plantas serão apresentados. A seção 3.1 é uma introdução ao capítulo. Na seção 3.2 é uma resposta a indagação sobre a inteligência das plantas. As seções seguintes são esboçados métodos bioinspirados nas plantas, como segue: seção 3.3 método de polinização das plantas; seção 3.4 método de colonização de ervas daninhas; seção 3.5 método de propagação da planta do morango; e por fim a seção 3.6 com o método do ciclo de vida da planta.

3.1. Introdução

O modo como as plantas sobrevivem e se adaptam em ambientes desfavoráveis tem despertado grande interesse de pesquisadores da inteligência artificial. É notável que o ciclo de vida de uma planta seja extremamente intrigante. A forma como se reproduzem, se propagam, dispersam suas sementes e selecionam as mais resistentes é sem dúvida uma forma inteligente de otimizar sua existência.

Este estilo de vida inteligente das plantas tem despertado interesse de alguns estudiosos que têm proposto algoritmos evolutivos inspirados no comportamento das plantas.

3.2. As plantas são inteligentes?

As plantas têm capacidade intrínseca para processar informações a partir de estímulos tanto abióticos como bióticos que permite decisões ótimas sobre as atividades futuras em um determinado ambiente [10]. Elas são capazes de perceber e responder otimamente a tantas variáveis ambientais - luz, água, gravidade, temperatura, estrutura do solo, nutrientes, toxinas, micróbios, herbívoros, sinais químicos de outras plantas - que pode se considerar a existência de algum sistema de processamento de informações

semelhante ao cérebro para integrar os dados e coordenar a resposta comportamental de uma planta [11].

A sobrevivência das espécies depende muitas vezes da sua capacidade de adaptação para encontrar comida rapidamente, evitar a predação e dar à sua prole a melhor chance de sobreviver e prosperar. Estes são tipicamente problemas de otimização/pesquisa [12]. As plantas têm uma capacidade de adaptação inigualável, o que garante a sobrevivência e a evolução de sua espécie mesmo em ambientes hostis. Comprovando que o comportamento adaptativo é projetado para melhorar a sobrevivência, reprodução e aptidão das espécies.

O ciclo de vida pode ser o principal objeto da seleção natural [13, 14]. O ciclo de vida das plantas evoluiu durante milhões de anos, tornando-as capazes de sobreviver em habitats com condições ambientais difíceis. Isto prova que as estratégias de sobrevivência das plantas são completamente eficazes na natureza [15]. As plantas evoluíram entre quinze a vinte sentidos distintos, incluindo análogos dos cinco sentidos dos humanos: cheiro e gosto (elas sentem e respondem a produtos químicos no ar ou em seus corpos); (elas reagem de forma diferente a vários comprimentos de onda de luz, bem como a sombra); toque (uma videira ou uma raiz "sabe" quando encontra um objeto sólido); e som (elas percebem/reagem a ondas sonoras) [11].

A planta recolhe e atualiza diversas informações sobre seus arredores, combina isso com informações internas sobre seu estado (raciocínio simples) e toma decisões que conciliam o seu bem-estar com o seu ambiente [16]. Várias partes das plantas se comunicam entre si através de tecidos e células, de maneira extraordinariamente complexa. A planta aprende por tentativa e erro, quando mudanças suficientes ocorrem ao seu redor elas agem de forma a minimizar o estresse e lesão interagindo e integrando as outras variáveis ambientais disponíveis [17]. A maneira mais simples de detectar se as plantas podem aprender, é colocá-las sob novas circunstâncias, ainda não experimentadas durante sua evolução e observar se elas conseguem acomodar e continuar o desenvolvimento [18].

A inteligência é definida como a capacidade de aprender ou compreender, e também inclui a capacidade de generalizar e adaptar, bem como habilidades de auto-organização, auto cura e auto reparação, qualquer sistema que tem pelo menos estas três habilidades é denominado como um sistema inteligente [19]. Há ampla prova de que as plantas possuem as características de inteligência [20].

De maneira simplificada, pode-se definir inteligência como a habilidade de resolver problemas e as plantas são seres incrivelmente boas para resolver seus problemas. Por

exemplo, para resolver a necessidade de energia a planta é adaptada para captar a maior quantidade possível de energia solar, absorvendo mais luz para fabricar energia. Outro incrível exemplo refere-se a algumas espécies de milho e feijão, para se proteger de taturanas, liberam uma substância no ar que atrai vespas. As vespas se aproximam e as taturanas, ao qual são presas das vespas, se vão. Com esse exemplo fica claro que a as plantas são capazes de aprender, comunicar, se adaptar e solucionar problemas de maneira inteligente. Portanto, é possível assegurar que as plantas são sim inteligentes.

3.3. Método de polinização das flores

O Algoritmo de Polinização das Flores (*Flower Pollination Algorithm* - FPA) foi inspirado na forma como as flores são polinizadas. O algoritmo foi proposto por Yang [21] e é aplicado em diversas áreas como: controle, engenharia elétrica [22, 23], resolução de jogos Sudoku [24], engenharia química [25]. O FPA também é utilizado conjuntamente com outros métodos IA de forma a ter métodos híbridos [24, 26], os quais conseguem melhorar a precisão de busca pelo ótimo [27].

3.3.1. Características da polinização das flores

Polinização é uma forma de reprodução das plantas floridas, ocorre através da transferência de pólens das anteras de uma flor para o estigma da mesma flor ou de flores de plantas da mesma espécie [28], como representado na Figura 3.1. Há dois tipos de polinização, a biótica e abiótica, como apresentado na Figura 3.2. Na polinização biótica, o pólen é transportado por um polinizador, um animal. Já a abiótica, o pólen depende de fatores ambientais para ser transportado. Dentro da polinização biótica há uma característica importante, a lealdade a uma flor (*flower constancy*), que é comportamento exibido pelos polinizadores que restringem as visitas em grande parte a um único tipo floral [29].

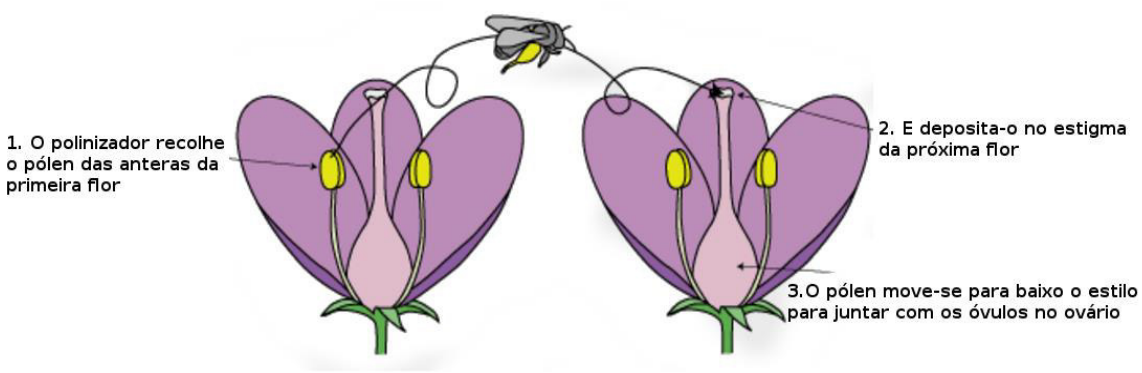

Figura 3.1 Processo de polinização.

Fonte: http://diyeverywhere.com/2016/09/15/what-you-need-to-know-about-pollination-for-a-robust-harvest/

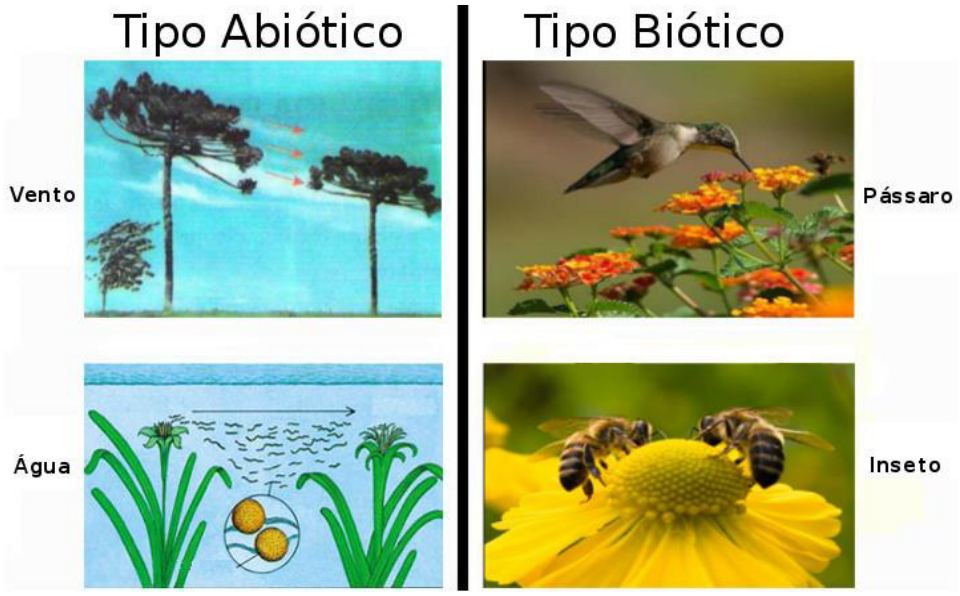

Figura 3.2 Tipos de polinização.

A polinização pode ser obtida através de dois mecanismos, são eles: autopolinização e polinização cruzada, tais mecanismos são representados na Figura 3.3. Na autopolinização, a reprodução de uma planta é garantida pelo transporte do pólen da sua própria flor ou pólen de flores da mesma planta para suas anteras. Já a polinização cruzada é garantida pelo transporte do pólen de uma flor para as anteras de uma flor de planta diferente.

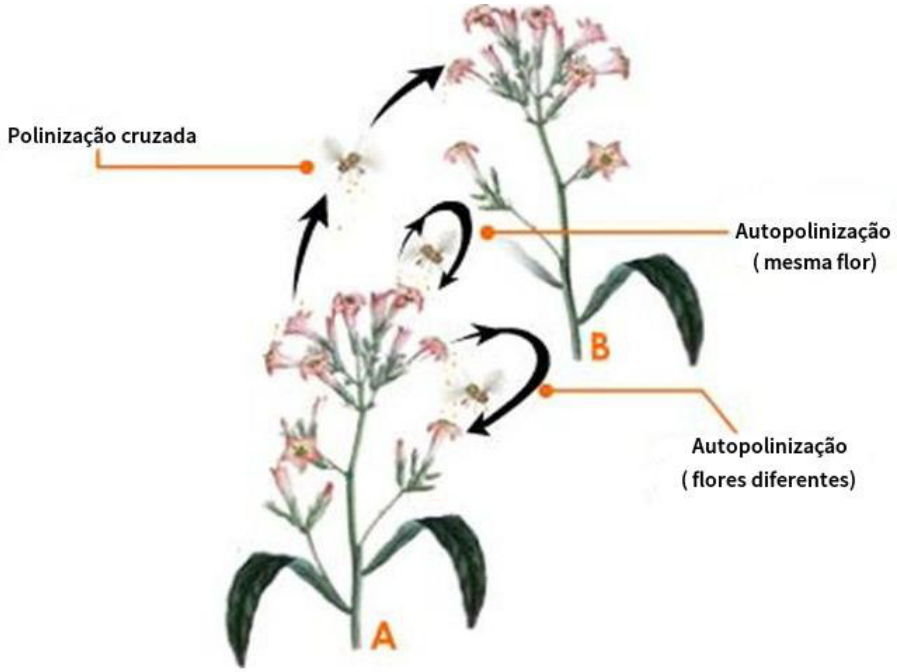

Figura 3.3 Mecanismos de polinização.

Fonte: http://biology.tutorvista.com/plant-kingdom/types-of-pollination.html

3.3.2. Algoritmo de polinização das flores

Todas essas características da polinização foram consideradas para elaboração do algoritmo. O algoritmo FPA obedece as seguintes regras:

- Polinização biótica e cruzada são consideradas como um processo de polinização global com os transportadores de pólen executando voos de Lévy [30];
- Polinização abiótica e autopolinização são consideradas como polinização local;
- Lealdade a uma flor pode ser considerada como tendo uma probabilidade de reprodução proporcional à similaridade das duas plantas envolvidas;
- Polinização local e global são controladas por uma probabilidade $prob \in [0,1]$. Devido à proximidade física e outros fatores como o vento, a polinização local pode ter uma significante parte nas atividades gerais de polinização.

Seguindo as regras especificadas, o algoritmo foi dividido em duas etapas, polinização local e polinização global. Considerando que só haja uma flor para cada planta e um pólen para cada flor, o algoritmo terá uma solução b_i.

Para a etapa de polinização global será encontrado o pólen mais apto, chamado g_*. O pólen mais apto é aquele que tem sua polinização e reprodução assegurada por suas características de sobrevivência aos transtornos do transporte, como a distância e o vento. Nessa etapa, será considerada a regra da lealdade a uma flor, representada matematicamente pela equação (1) [21]:

$$b_i^{t+1} = b_i^t + L(b_i^t - g_*) \tag{1}$$

Onde b_i^t é o pólen i ou vetor solução b_i na iteração t, e g_* é a melhor solução atual encontrado entre todas as soluções. O parâmetro L é o tamanho do passo.

Na polinização biótica, sendo o mais comum tipo de polinização que ocorre na natureza, os organismos que visitam as flores para coletar pólen, néctar, óleos ou mesmo odores, são os responsáveis pela transferência de grãos de pólen das anteras de uma flor para o estigma da mesma flor ou de uma outra flor da mesma espécie. Para demonstrar o movimento (voos) destes organismos, tais como abelhas, vespas, borboletas, mariposas e formigas, pode-se utilizar a distribuição de Lévy. Matematicamente, a distribuição de Lévy foi representada por (2) [21]:

$$L \sim \frac{\lambda \Gamma(\lambda) sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, (s \gg s_0 > 0) \tag{2}$$

Em que, $\Gamma(\lambda)$ é a função gama padrão válida para $s > 0$.

A segunda etapa representa a polinização local, ao qual é o modo incomum de polinização, que ocorre em apenas 10% nas plantas sem a ajuda dos animais. Matematicamente, este tipo de polinização pode ser representado por (3) [21].

$$b_i^{t+1} = b_i^t + \epsilon\left(b_j^t - b_k^t\right) \tag{3}$$

Em que, b_j^t e b_k^t imitam a lealdade a uma flor em um espaço limitado. Os índices j e k são escolhidos aleatoriamente (no caso utilizou-se a função *randperm* do MatLab) e pode ser comparada a operação de mutação do algoritmo Evolução Diferencial (ED). O parâmetro ϵ é um número aleatório que possui uma distribuição uniforme [0,1].

A Figura 3.4 ilustra o fluxograma do algoritmo de polinização de flores, onde N é o número de plantas, N_iter é o número máximo de iterações, t é o contador de iterações,

tol é uma tolerância para encontrar a melhor solução dada pelo absoluto f_{min}. O termo f_{min} refere-se a melhor solução encontrada. O algoritmo FPA é interessante, pois apresenta poucos parâmetros de entrada em comparação com outros clássicos algoritmos de IA. Por exemplo, o algoritmo PSO, apresenta os parâmetros c_1 e c_2, além de outros parâmetros. No caso do FPA, somente os parâmetros da distribuição de Lévy são necessários e o termo $prob$ que controla o tipo de polinização. Um número aleatório uniforme é gerado e o termo $prob$ controla o tipo de polinização. Este termo, é tipicamente maior do que 70% visto que a maior parte da polinização que ocorre na natureza é do tipo global.

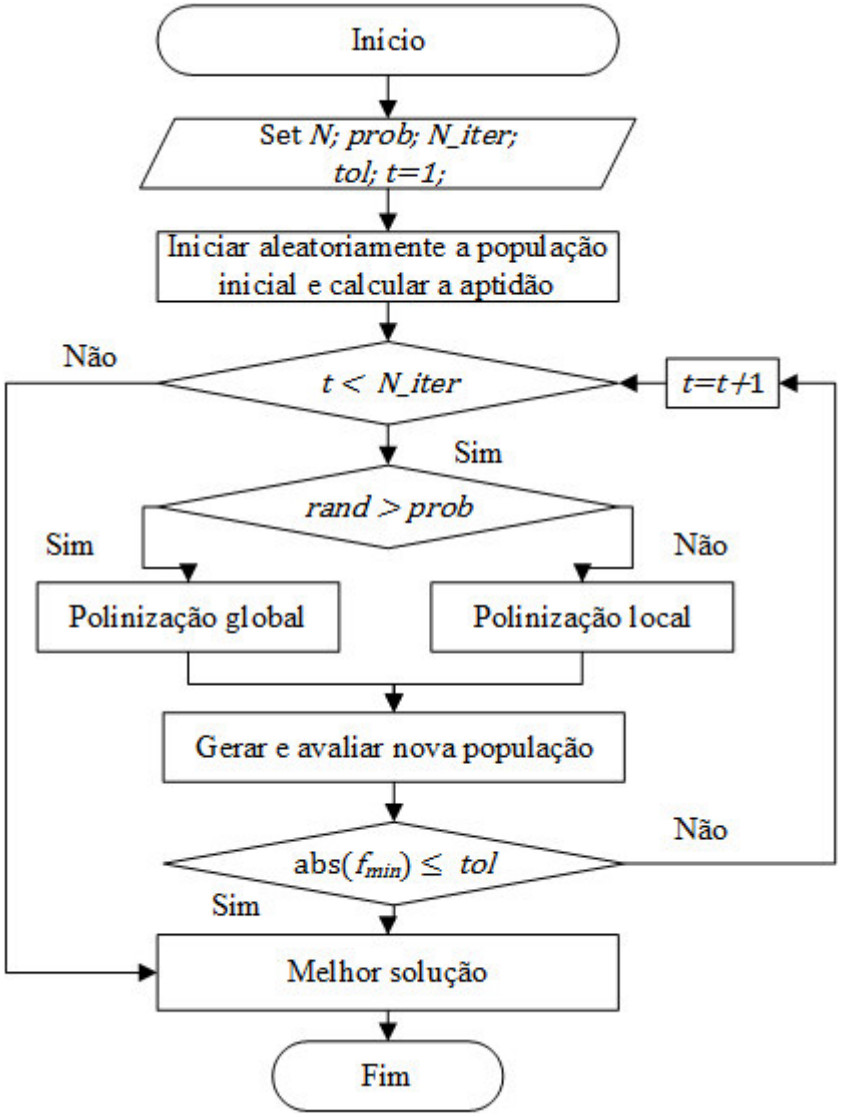

Figura 3.4 Fluxograma do algoritmo de polinização de flores.

3.4. Método de colonização de ervas daninhas

O algoritmo de colonização das ervas daninhas (*Invasive Weed Optimization* - IWO) é um algoritmo de otimização proposto por Mehabian [31] e utilizado para configuração de antenas [32, 33], resolução de problemas eletromagnéticos [34], resolução de sistemas não lineares [34]. O método é baseado em um fenômeno comum na agricultura, a colonização de ervas invasoras.

3.4.1. Características da colonização da erva daninha

Sendo considerada como ervas invasoras (daninhas) todas as plantas que crescem em locais onde não são desejadas, tornando-as ameaça as plantas que foram cultivadas [31]. As plantas daninhas são resistentes e capazes de se adaptar aos ambientes, mudando seu comportamento. Novas espécies de ervas daninhas aparecem frequentemente, mesmo com uso de herbicidas. A Figura 3.5 mostra a existência de buva (planta invasora) em uma plantação de soja.

Figura 3.5 Planta daninha buva em uma plantação de soja.

Fonte: http://ragricola.com.br/wp-content/uploads/2015/10/Buva.jpg

A forma como as plantas daninhas se reproduzem depende do tipo da planta, podendo ser de forma sexuada ou assexuada. A reprodução sexuada de plantas floridas ocorre através da polinização. A planta produz flores em determinada época, essas flores são responsáveis pela reprodução, há a geração de sementes que serão dispersas por polinizadores animais ou fatores ambientais, há uma "competição" entre as sementes e uma seleção das melhores (as que sobreviverão), essas sementes que sobrevirem se tornarão novas plantas.

As plantas daninhas são caracterizadas pelo seu alto poder de disseminação, o que favorece o seu crescimento em lugares indesejáveis. A competição das plantas daninhas com outras plantas ocorre principalmente devido à sua agressividade e grande produção de sementes com altas capacidades de disseminação e longevidade. Outros fatores que também caracterizam algumas espécies de plantas daninhas são as suas exigências fisiológicas relativamente baixas, as altas taxas de crescimento e as elevadas tolerâncias às variações ambientais [36].

3.4.2. Algoritmo da colonização da erva daninha

A elaboração desde algoritmo considera a forma de reprodução sexuada de plantas floridas. Para simular computacionalmente o comportamento de uma erva daninha serão levadas em consideração três etapas do ciclo de vida de uma planta, são elas: reprodução, dispersão de sementes e exclusão competitiva.

Na etapa da reprodução, cada membro da população de plantas está autorizada a produzir sementes em função da sua própria, menor e maior, aptidão da colônia, tendo o número de sementes produzidas linearmente de acordo com sua aptidão. Esta forma de reprodução garante que indivíduos inviáveis (com alto número de aptidão) tenha a chance de sobreviver e reproduzir, pois acredita-se que é possível que alguns dos indivíduos inviáveis transporte informação mais útil do que os indivíduos viáveis durante o processo de evolução.

A etapa de dispersão de sementes considera a forma de dispersão espacial, em que, as sementes geradas estão sendo distribuídas aleatoriamente através do espaço de busca dimensional d, por números aleatórios normalmente distribuídos com média igual à zero, mas variando a variância. No entanto, o desvio padrão (SD, do inglês *Standard Deviation*), σ, da função aleatória, será reduzido a partir de um valor inicial $\sigma_{\text{inicial}}(Si)$ e

um valor final $\sigma_{\text{inicial}}(Sf)$ predefinidos. A equação (4) [31] representa a forma de encontrar o desvio padrão de acordo com a iteração atual (σ_{iter}).

$$\sigma_{iter} = \frac{(iter_{max} - iter)^n}{iter_{max}}\left(\sigma_{inicial} - \sigma_{final}\right) + \sigma_{final} \tag{4}$$

Onde $iter_{max}$ é o número máximo de iterações, σ_{iter} é o SD no presente passo tempo e n é o índice de modulação não linear (geralmente igual a 3).

A etapa de exclusão competitiva vai garantir que os melhores indivíduos da população sobrevivam e formem a nova população. Quando o número máximo de sementes for atingido, a população é avaliada e ordenada de acordo com seu *fitness*, o indivíduo com melhor *fitness* é selecionado e formará a nova população. Todo o processo acontece até que o número máximo de iteração seja atingido.

A Figura 3.6. ilustra o fluxograma de IWO. O algoritmo irá iniciar uma população com um número mínimo de plantas N_{min}, essa população será avaliada por uma função objetivo, e então começam os processos evolutivos da planta, os quais são repetidos até um número máximo de iteração N_iter, ou até ser atingida uma condição de parada que encontrará o mínimo da função através de uma tolerância, tol. Dentro do laço repetitivo, a população inicial será ordenada de forma crescente de acordo com o seu *fitness* (aptidão) calculado, e de acordo com suas respectivas aptidões elas produzem sementes, etapa da reprodução. Após a produção de sementes, as mesmas são dispersas aleatoriamente sobre o espaço de busca, elas crescem e se tornam novas plantas, esse processo continua até que um número máximo de planta N seja atingido. As novas plantas serão avaliadas, ordenadas de forma crescente de acordo com seu *fitness*, e apenas as melhores plantas sobreviverão e gerarão novas sementes, representando o processo de exclusão.

Uma importante propriedade do algoritmo IWO é que todos as plantas participam do processo de reprodução. Plantas com altos *fitness* produzem mais semente comparadas com aqueles de menor *fitness*. Esta característica melhora a convergência do algoritmo. Além disso, é possível que algumas plantas com baixo *fitness* carreguem importantes informações comparadas com as de alto *fitness*.

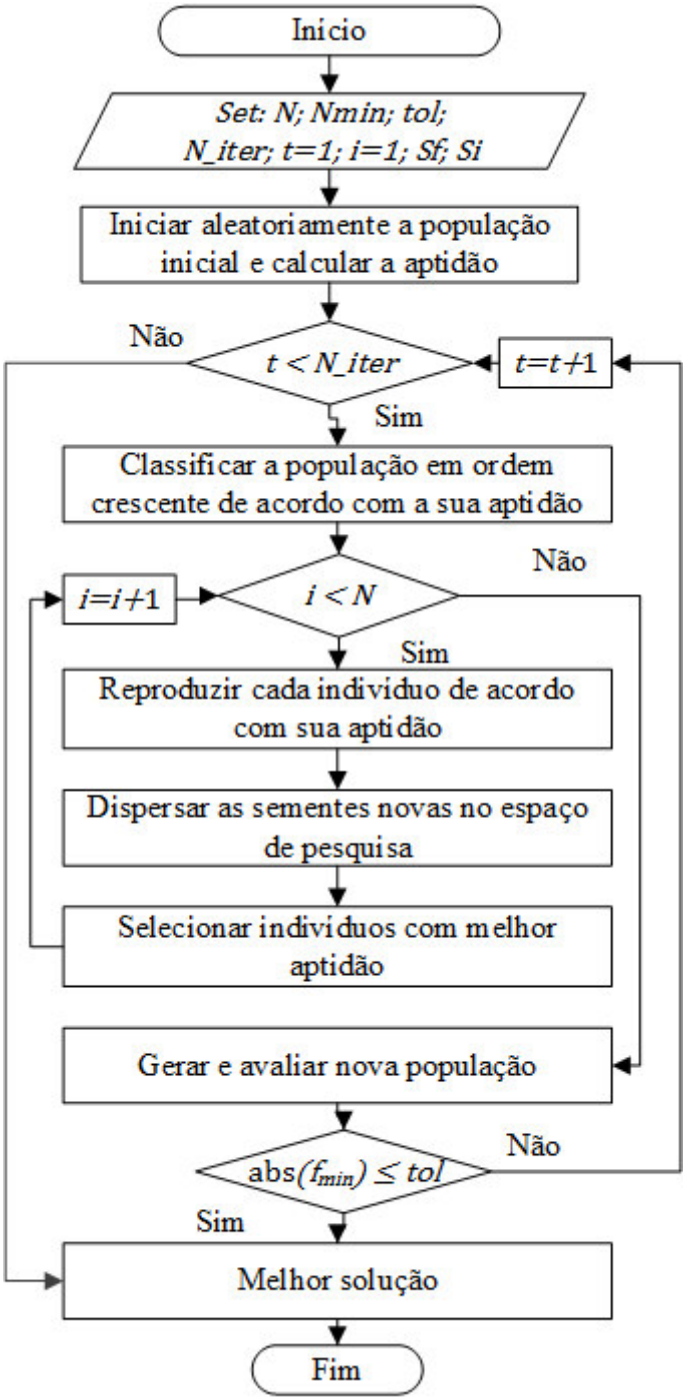

Figura 3.6 Fluxograma do algoritmo colonização de ervas daninhas.

3.5. Método de propagação da planta do morango

O Algoritmo do Morango (*Strawberry Algorithm*-SBA) proposto por Merrikh-Bayat [37], é um algoritmo de otimização numérica inspirado na planta do morango para resolver problemas multivariáveis, tendo como característica o número de agentes

computacionais não sendo constante do começo ao fim. Em cada iteração, o número de agentes computacional é duplicado, de forma adequada e em seguida metade dos agentes mais fracos é eliminada.

3.5.1. Características de propagação da planta do morango

A planta do morango é um tipo especial de planta, pois é capaz de se movimentar sozinha, ou seja, sua raiz não está presa a terra como a maioria das outras plantas, como representada na Figura 3.7. A planta do morango consegue se propagar através de corredores ou estolão em busca de nutrientes e melhores condições para sua sobrevivência. A Figura 3.8 ilustra o comportamento da planta do morango.

Figura 3.7 Planta do morango.

Fonte: http://pt.depositphotos.com/11637521/stock-photo-strawberry-plant.html

Corredor é um caule rastejante produzido nas axilas das folhas e cresce para fora da planta mãe. Inicialmente, ao longo do corredor vão se criando novos nós e desses nós surgem às plantas filhas que ainda não tem raízes, mas assim que vão crescendo adquirem raízes fibrosas e já estão aptas a se separarem das plantas mães e continuarem suas vidas individualmente gerando novas plantas filhas.

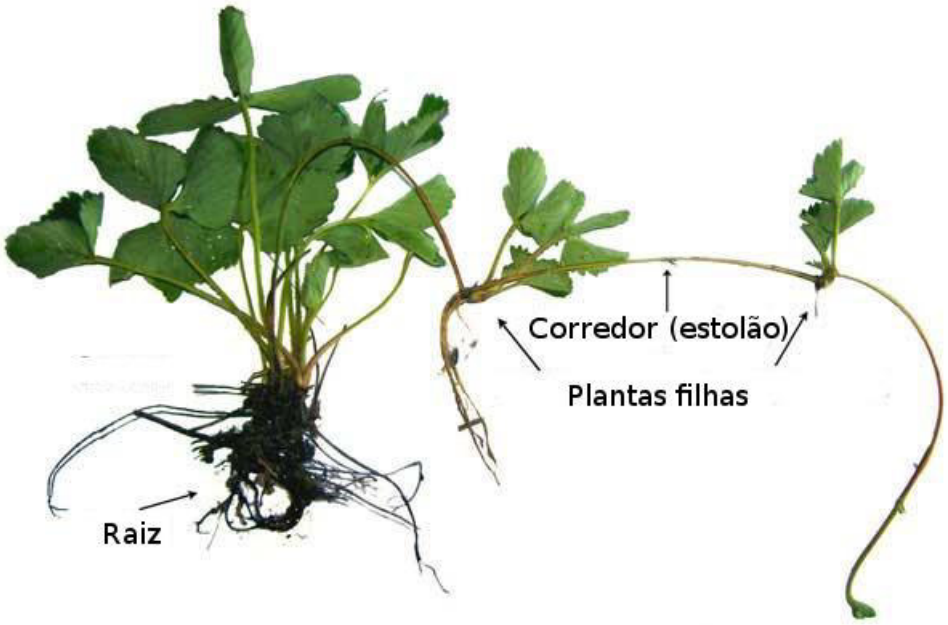

Figura 3.8 Comportamento da planta do morango.

Fonte: http://strawberryplants.org/2010/05/strawberry-plant/

3.5.2. Algoritmo da propagação da planta do morango

Baseado nas características da reprodução da planta do morando, o algoritmo SBA assume que no domínio do problema são gerados certos números N de agentes computacionais, tais agentes são considerados plantas sejam mães ou filhas, e a cada iteração *t* do algoritmo essas plantas geram corredores e raízes, perto e longe de si [37]. O algoritmo é dividido em duas etapas de busca pela melhor solução, duplicação e eliminação.

Em SBA assume-se que, $x_j(t)\ \epsilon\ R^m$ está para a localização da planta mãe $jth\ (j = 1,2,\dots,n)$ para $t-th$ iteração, a matriz contendo a localização correspondente a corredores e raízes nessa iteração, $X_{pop}(t)$, e calculada por (5)-(7) [37]:

$$X_{prop}(t) = [X_{root}(t)X_{runner}(t)] = [X(t)X(t)] + [d_{root}r_1 d_{runner}r_2] \quad (5)$$

$$X(t) = \left[x_{1,root}(t)n_{2,root}(t)\dots x_{N,root}(t)\right] \quad (6)$$

$$X_{runner}(t) = \left[X_{1,runner}(t)\ X_{2,runner}(t) \ldots X_{N,runner}(t)\right] \quad (7)$$

$r_1, r_2 \in R^{m*N}$ são matrizes aleatórias cujas entradas são números aleatórios independentes com distribuição uniforme no intervalo [0.5,-0.5], $L_{root}\ e\ L_{runner}$ são dois escalares que representam a distância de raízes e corredores da planta mãe, e N é o tamanho de planta mãe.

De acordo com a equação (5), cada planta mãe gera dois pontos, um mais perto e um mais longe. Assim, a aplicação de (5) duplica o número de agentes, isto é, $X_{prop}(t)$ tem colunas $2N$, enquanto $X(t)$ só tem N colunas.

Após o cálculo de $X_{prop}(t)$, usa-se um esquema de seleção para selecionar as colunas N de $X_{prop}(t)$ denotado como $fit\left(x_{j,prop}(t)\right)$, através de (8) [37].

$$fit\left(x_{j,prop}(t)\right) = \begin{cases} \dfrac{1}{a + f\left(x_{j,prop}(t)\right)} & f\left(x_{j,prop}(t)\right) > 0 \\ a + \left|f\left(x_{j,prop}(t)\right)\right| & f\left(x_{j,prop}(t)\right) \leq 0 \end{cases}, j = 1, \ldots N \quad (8)$$

Onde, $f(.)$ é a função custo para ser minimizada. Após calcular os valores de aptidão, a escolha dos $j - th$ na coluna, p_j é calcula de acordo com a regra de seleção por roleta.

A Figura 3.9 ilustra o fluxograma do funcionamento do algoritmo SBA. Na etapa de duplicação a planta mãe gera dois pontos aleatórios (plantas filhas), um perto de si e um mais distante de si, que modelam as raízes (L_root) e corredores (L_runner). As raízes fazem à busca local e os corredores a busca global. Mais precisamente, a planta mãe recebe dois vetores coluna. A planta duplicada é então avaliada de acordo com uma função objetivo (*fitness*) e então começa a etapa de eliminação, que de acordo com o valor de aptidão resultante há uma seleção dos melhores vetores (plantas) que sobreviverão para formar a nova população, e os menos aptos serão eliminados. A população retorna então ao seu tamanho inicial. Este procedimento se repete até que o numero máximo de iteração N_iter seja atingido ou se a melhor solução seja encontrada a partir de uma condição de tolerância tol.

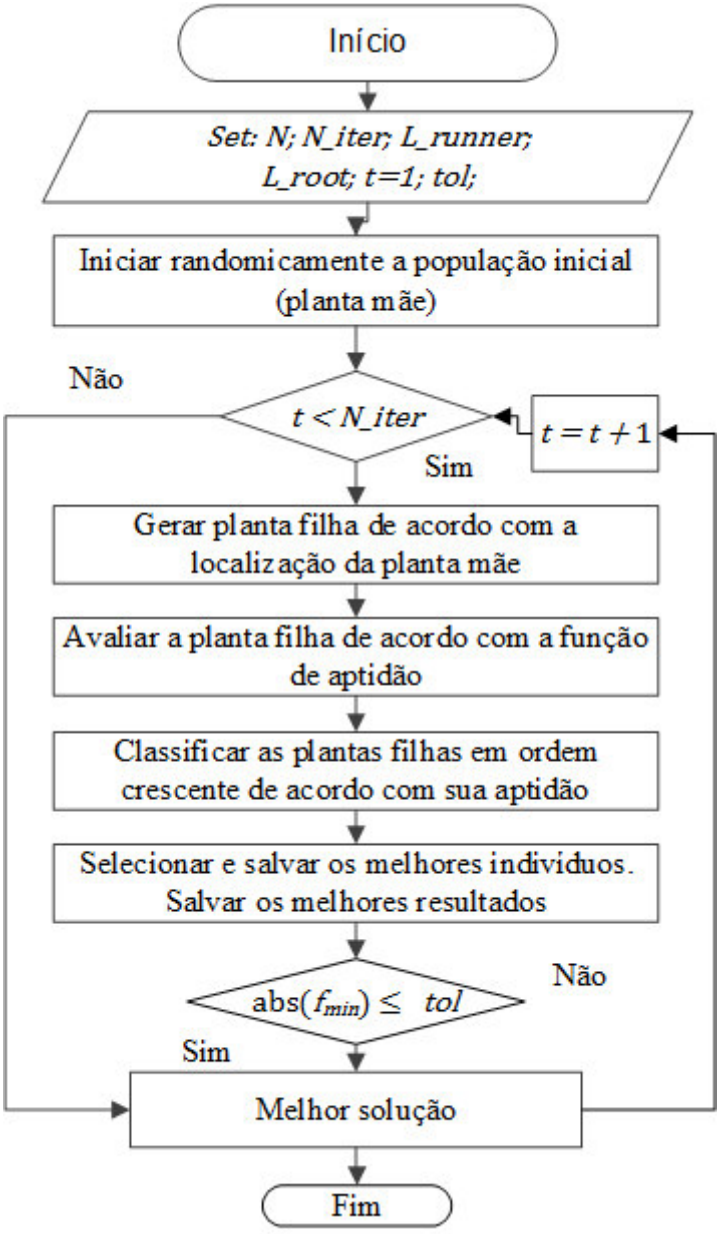

Figura 3.9 Fluxograma do algoritmo da planta do morango.

3.6. Método do ciclo de vida da planta

O novo algoritmo bioinspirado foi proposto por Karami [38]. É inspirado no ciclo de vida das plantas (*Plant Life Cycle* - PLC). As plantas evoluem durante os anos e tornam-se capazes de sobreviver em habitats com condições ambientais difíceis. Apesar de não poderem se mover, através da evolução elas encontraram meios inovadores para encontrar locais adequados para crescer e se reproduzir. O PLC considera as fases do ciclo

de vida real das plantas e emprega-as de forma artificial para achar uma resposta ótima no espaço de busca. Utilizam a pesquisa local e global para encontrar a resposta ideal.

3.6.1. Características do método do ciclo de vida da planta

As plantas estão entre as primeiras criaturas da Terra, estando presentes em quase todos os ambientes do planeta, até mesmo onde a quantidade de luz, água e nutrientes são escassos. Elas evoluem de acordo com o ambiente em que estão [38].

Há diferentes tipos de reprodução de plantas, a mais comum, e a que será considerada nesse trabalho, é a reprodução por produção e dispersão de sementes. Considerando que em épocas especificas as plantas produzem flores, sendo as flores responsáveis por sua reprodução. A Figura 3.10 ilustra o ciclo de vida de uma planta.

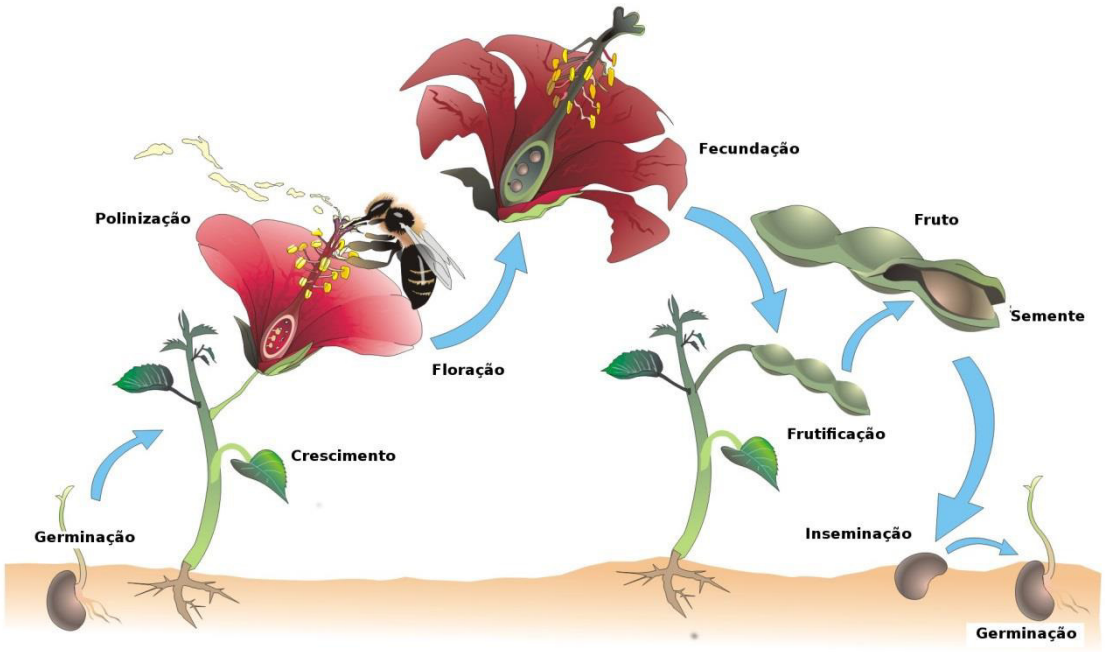

Figura 3.10 Ciclo de vida de uma planta florida.

Fonte: https://proyectoeducere.wordpress.com/tag/reproduccion-sexual/

A reprodução acontece na forma da polinização. A polinização é o transporte de pólen das anteras para o estigma das plantas, seja dela mesma (autopolinização) ou de outra planta da mesma espécie (polinização cruzada), por um polinizador. Antera e estigma

são os órgãos sexuais das plantas. O processo de encontro do esperma (presente nas anteras) com o óvulo (presente no estigma) é chamado de fertilização. Os óvulos fertilizados produzem sementes que serão a próxima geração de plantas [21].

Por terem mobilidade limitada, as plantas dependem de diversos fatores para a polinização e dispersão de sementes. Essa dispersão tem um papel importante na manutenção da diversidade de espécies, ela permite que as plantas alcancem lugares mais favoráveis para sua sobrevivência [12].

Outro ponto a se destacar é a sobrevivência das sementes, não só no instante da dispersão, mas também na competição com outras sementes e plantas maiores por melhores condições de vida e localização. Considerando que elas precisam de bastante nutrientes, água e luz para sobreviverem. Desta forma percebe-se que há uma seleção natural, sobrevivendo apenas às sementes e plantas que estão em habitats mais favoráveis às suas condições [14,18].

3.6.2. Algoritmo do ciclo de vida da planta

O algoritmo PLC aborda cinco principais etapas do processo de ciclo de vida de uma planta, são elas: inicialização, polinização, fertilização, dispersão de sementes e seleção de sementes em relação à concorrência local.

A primeira etapa do algoritmo, inicialização, se dar pela geração aleatória de uma população inicial N de plantas, além da avaliação de aptidão dessa população inicial através de uma função objetivo, que neste trabalho será abordada como uma função *fitness* e a classificação dos indivíduos da população (plantas) de acordo com sua aptidão. A segunda etapa é a polinização, realizada de forma aleatória na população, sobre a qualidade de seus indivíduos e a distância entre eles. A distância entre as plantas pode ser calculada por (9) [38]:

$$P_{dis}(i,j) = \frac{1}{\sum_{k=1}^{n} \frac{dis(i,j)}{dis(i,k)}} \tag{9}$$

Onde, $dis(i,j)$ é a distância entre $i-th$ e $j-th$ da planta.

A qualidade da planta pode ser calculada por (10) [38]:

$$P_c(i) = \frac{C_{max} - C(i)}{\sum_{k=1}^{n}(C_{max} - C(k))} \tag{10}$$

Em que, C_{max} é o custo máximo da população de plantas (pior qualidade da planta), enquanto $C(i)$ custo do i da planta.

Há sempre uma possibilidade para cada flor receber múltiplos pólens, essa possibilidade é aleatória e representada por P_n. No entanto, apenas o pólen de melhor qualidade vai fertilizar o óvulo, segundo (11) [38].

$$N_{polen}(i) \sim N(\mu = P_n, \sigma = 0.5P_n) \tag{11}$$

A próxima etapa é a fertilização, processo em que a semente é produzida. Cada planta produz varias sementes em cada ciclo de reprodução, e um fator P_s, determina o número médio de sementes produzidas por cada planta. A fertilização é realizada utilizando algum tipo de cruzamento genético.

A quarta etapa é a dispersão de sementes. As sementes são dispersas usando uma função de distribuição normal com média de posição atual da planta mãe e variância especificada. Variância de dispersão é calculada baseada na variância do k vizinhos mais próximos da planta mãe em cada etapa. Para ajustar a variância de dispersão o parâmetro P_w é utilizado. P_w mostra a porcentagem de iterações em que cada variância de dispersão é ajustada para um valor grande em comparação com a variância normal da dispersão.

O último passo é a seleção. A seleção vai depender da qualidade de custo da planta. A planta vai competir com seus k vizinhos mais próximos, eles são avaliados pelo *fitness* e então o de melhor qualidade é selecionado.

O algoritmo PLC proposto é resumido como segue na Figura 3.11:

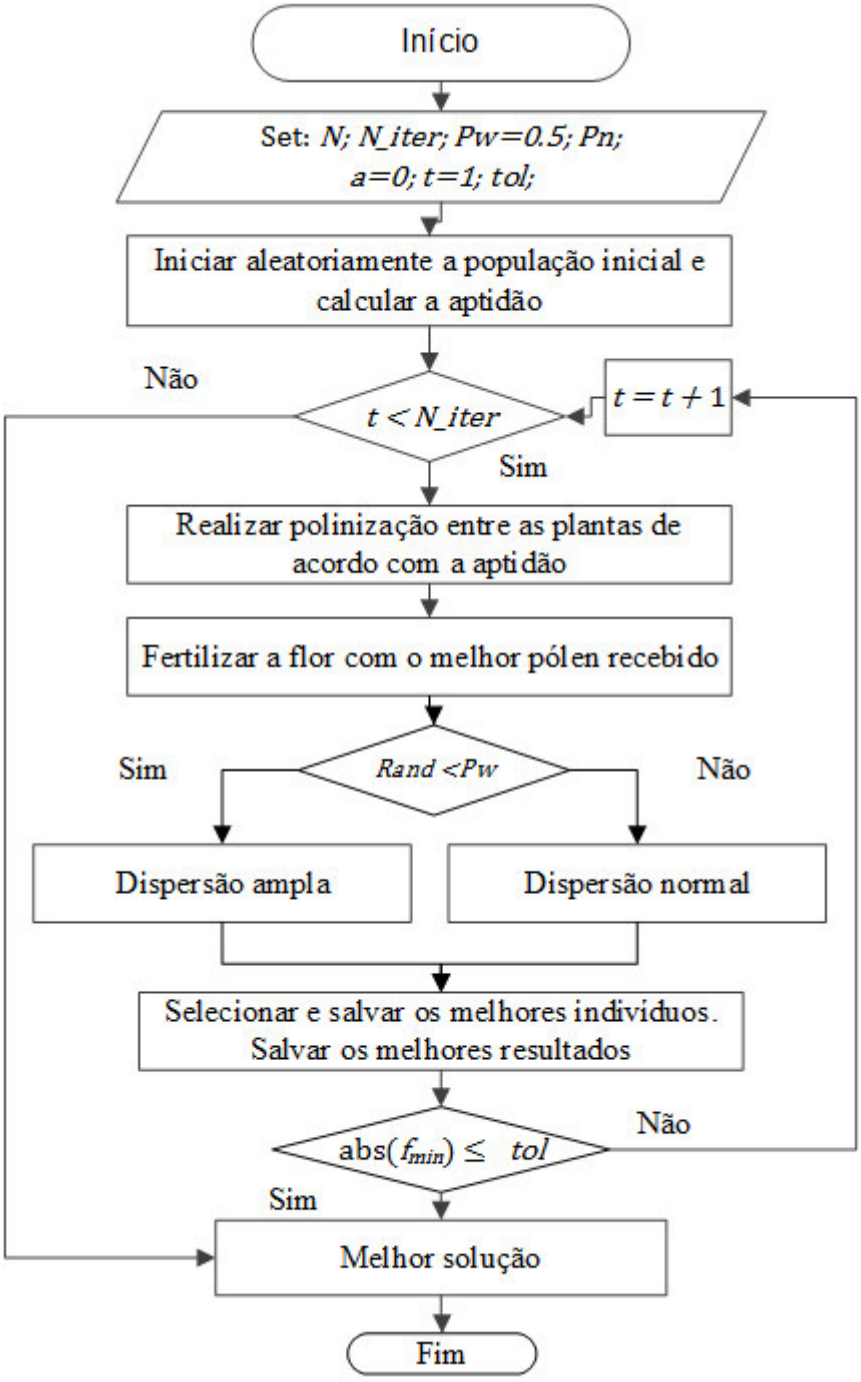

Figura 3.11 Fluxograma do algoritmo ciclo de vida da planta.

.

Capítulo 4

4. Performance dos algoritmos inspirados na inteligência das plantas

Para testar o desempenho dos algoritmos aqui apresentados, um conjunto de funções teste uni-modais, multimodais e multimodais com dimensão fixas foram usadas, que são aplicadas a problemas de minimização de acordo com [39]. A seção 4.1 é uma introdução ao capítulo. Na seção 4.2 são apresentadas as funções benchmark utilizadas para testar os algoritmos. A seção 4.3 traz a performance dos métodos testados pelas funções apresentadas na seção 4.2.

4.1. Introdução

A todo momento surgem novos algoritmos de otimização e não se sabe o quão eficiente esse novo método é, e se o mesmo é a melhor solução para o problema ao qual foi submetido a resolver. Para definir a eficiência dos métodos são utilizadas funções de teste, chamadas de funções benchmark, estas funções têm extremos locais, podem testar a capacidade do algoritmo em encontrar extremo global em diferentes dimensões [38]. as funções de testes conhecidas e utilizadas na literatura são inúmeras, no entanto não existe um número acordado de funções para testar novos algoritmos [40, 41, 42].

4.2. Funções benchmark

Para teste de confiabilidade, eficiência e validação de algoritmos de otimização são frequentemente utilizados um conjunto escolhido de padrões de referência ou funções de teste da literatura. As funções teste são problemas artificiais e podem ser usadas para avaliar o comportamento de um algoritmo em situações diversas e às vezes difíceis. Problemas artificiais podem incluir mínimos globais, mínimos locais, vales longos e estreitos, efeitos de espaço nulo e superfícies planas. Estes problemas podem ser facilmente manipulados e modificados para testar os algoritmos em diversos cenários [40].

Funções benchmark podem ser classificadas em termos de características como modalidade (refere-se aos picos ambíguos na paisagem de uma função, se durante o processo de busca os algoritmos encontrarem esses picos há a possibilidade de ficarem

presos neles e não encontrarem as verdadeiras soluções ótimas), bacias (declínios íngremes em torno de uma grande área [43]), vales (ocorrem quando uma área estreita de pouca mudança é cercada por regiões de descida íngreme [43]), separabilidade (medida da dificuldade de diferentes funções) e dimensionalidade (está relacionado com a dimensão do espaço de característica da função) [44].

Neste trabalho, algumas funções de teste benchmarks de acordo com as propriedades de modalidade e dimensionalidade serão abordadas. As funções uni-modais (com apenas um ótimo local) são apresentadas na Tabela 4.1 [42], as funções multimodais (com mais de um ótimo local) são apresentadas na Tabela 4.2 [42] e as funções multimodais com dimensão fixas são representadas na Tabela 4.3 [42].

Tabela 4.1 Funções teste uni-modais

Funções	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
$f_5(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \right]$	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

Tabela 4.2 Funções teste multimodais

Funções	Dim	Range	f_{min}
$f_8(x) = \sum_{i=1}^{n} -x_i sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	[-500,500]	-12569,5
$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i) + 10]$	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
$f_{12}(x) = \frac{\pi}{n}\left\{10sin(\pi y_i) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	[-50,50]	0
$f_{13}(x) = 0.1\left\{sin^2(3\pi x_i) + \sum_{i=1}^{n}(x_i - 1)^2[1 + sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + sin^2(2\pi x_n + 1)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	[-50,50]	0

Tabela 4.3 Funções teste multimodais com dimensão fixa

Funções	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
$f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_i(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	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
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)cosx_1 + 10$	2	[-5,5]	0.398
$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	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

4.3. Performance dos métodos

Os algoritmos FPA, IWO, SBA e PLC apresentados no Capítulo 3 foram testados com as mesmas funções, apresentadas na seção 4.2, e comparados entre si e com o algoritmo genético clássico (AG). Seguindo seus respectivos fluxogramas para suas implementações, destaca-se as características e particularidades de cada algoritmo, observando sempre seus parâmetros inicias.

Para testar os algoritmos foi utilizado a plataforma MATLAB (Matrix Laboratory), em um PC DELL Inspiron 15, com configuração: processador Intel(R) Core(TM) i7-

5500U CPU @ 2,40GHz, memória RAM 8,00GB, sistema operacional Windows 10 Home Single Language de 64 bits, placa de vídeo gráfica AMD.

Os parâmetros individuais foram definidos da seguinte forma: para algoritmo FPA usou-se uma probabilidade $Prop = 0.8$; para o algoritmo SBA foram definidos como tamanhos de corredores $L\_runner = 40$ e raízes $L\_root = 10$. Já para o algoritmo IWO define-se que o número mínimo de plantas $Nmin = 10$, $S_{\mathrm{i}} = 0{,}5$ e $S_{\mathrm{f}} = 0{,}001$, valores inicial e final do desvio padrão, respectivamente. Para o AG foi utilizada taxa de mutação $tm = 0{,}05$ e taxa de crossover $tc = 0{,}95$.

Para todos algoritmos alguns parâmetros foram definidos igualmente, como: o número máximo de iterações $N\_iter = 1000$, o tamanho da população N = 20, 50 e 100, sendo que cada algoritmo foi testado com cada um desses valores de população, a tolerância $tol = 10^{-05}$, e todos os algoritmos foram testado em cada função 10 vezes, achando sua melhor solução.

As Tabelas 4.4, 4.5 e 4.6 mostram os melhores resultados obtidos para cada algoritmo testado com as funções de referência apresentadas nas tabelas da seção 4.2.

Tabela 4.4 Melhores soluções encontradas para População = 20

		MÉTODOS				
Funções	f_{min}	**FPA**	**SBA**	**IWO**	**PLC**	**AG**
f_1	0	1.70e-05	2.06e-08	2.89e-08	0.0501	2.24e-07
f_2	0	0.0000	3.01e-07	1.90e-06	9.34e-04	7.39e-07
f_3	0	2.11e-06	2.80e-07	5.87e-08	0.7916	3.71e-07
f_4	0	1.48e-04	4.17e-07	2.96e-08	0.5754	1.64e-06
f_5	0	6.14e-05	7.40e-09	1.66e-07	0.0000	4.50e-07
f_6	0	4.67e-08	1.10e-06	3.14e-08	0.0489	2.27e-08
f_7	0	3.38e-06	1.64e-07	3.44e-06	1.80e-05	7.07e-06
f_8	-12569,5	-12569	8.47e-08	-1,26E+04	-1,25E+05	-1,26E+04
f_9	0	3.83e-05	5.38e-07	3.07e-09	0.0295	7.92e-03
f_{10}	0	1.65e-04	7.87e-10	8.51e-07	0.2971	1.81e-06
f_{11}	0	6.24e-05	2.94e-09	5.20e-08	0.2131	4.77e-08
f_{12}	0	1.50e-08	1.65e-07	6.94e-07	2.63	1.25e-07
f_{13}	0	3.22e-06	1.91e-07	1.19e-07	0.0715	1.08e-07
f_{14}	1	0.99	1.45e-07	0.9980	0.9980	0,998
f_{15}	0.00030	0.0000	5.82e-08	0.0017	0.0024	0.00
f_{16}	-1.0316	4.78e-09	6.06e-08	5.55e-09	0.9948	-1,02E+00
f_{17}	0.398	0.39	7.71e-09	17.17	0.4020	17.17
f_{18}	3	32.68	1.11e-06	32.68	4.56	32.68
f_{19}	-3.86	-1,89	4.32e-08	-1,8997	-1,89	-1,89
f_{20}	-3.32	-1,16E+00	3.26e-08	-1,16E+00	-1,16	-1,16
f_{21}	-10.1532	-10,15	4.86e-08	-10,15	-10,15	-10,15
f_{22}	-10.4028	-10,4	3.16e-08	-10,4	-10,4	-10,4
f_{23}	-10.5363	-10,53	1.76e-07	-10,53	-10,52	-10,53

Tabela 4.5 Melhores soluções encontradas para População = 50

		MÉTODOS				
Funções	f_{min}	**FPA**	**SBA**	**IWO**	**PLC**	**AG**
f_1	0	7.93e-05	2.73e-08	2.80e-09	0.0276	5.28e-08
f_2	0	1.53e-04	1.78e-09	1.57e-06	0.0228	1.89e-06
f_3	0	2.41e-04	3.32e-07	6.72e-10	0.0508	4.51e-07
f_4	0	0.0000	8.16e-09	2.23e-07	0.2022	3.71e-06
f_5	0	1.46e-06	7.39e-07	9.63e-08	0.0000	3.35e-09
f_6	0	8.30e-06	1.61e-07	6.96e-07	0.0140	7.87e-08
f_7	0	3.39e-06	2.39e-08	1.57e-06	1.90e-05	5.47e-06
f_8	-12569,5	-12500	1.74e-08	-12569	-1,12e+05	-1,26e+04
f_9	0	3.93e-07	4.83e-09	1.49e-07	3.49e-05	6.23e-08
f_{10}	0	2.69e-04	1.20e-09	1.70e-06	0.0684	7.20e-06
f_{11}	0	1.05e-05	2.78e-08	2.95e-09	0.0502	1.45e-08
f_{12}	0	1.50e-10	6.83e-10	2.46e-06	0.3322	2.80e-10
f_{13}	0	1.68e-05	2.47e-08	3.95e-11	0.0026	2.67e-07
f_{14}	1	0.99	5.51e-09	0.9980	0.9980	0.9980
f_{15}	0.00030	0.00	1.21e-07	0.0017	0.0057	7.93e-04
f_{16}	-1.0316	5.16e-07	9.61e-13	5.41e-08	-1,03E+00	-1,00E+0
f_{17}	0.398	0.84	6.90e-09	17.17	0.4666	17.17
f_{18}	3	32.68	1.29e-07	32.68	4.16	31.57
f_{19}	-3.86	-1,89	8.89e-09	-1,9897	-1,89	-1,89
f_{20}	-3.32	-1,16E+00	7.25e-08	-1,16	-1,16	-1,16
f_{21}	-10.1532	-10,15	2.86e-08	-10,15	-10,15	-10,15
f_{22}	-10.4028	-10,4	1.50e-07	-10,4	-10,4	-10,4
f_{23}	-10.5363	-10,53	2.91e-07	-10,53	-9,98	-10,53

Tabela 4.6 Melhores soluções encontradas para População = 100

		MÉTODOS				
Funções	f_{min}	**FPA**	**SBA**	**IWO**	**PLC**	**AG**
f_1	0	3.21e-05	5.34e-08	1.64e-05	0.1579	3.60e-07
f_2	0	7.38e-05	1.28e-07	0.0171	0.0237	1.42e-06
f_3	0	5.47e-05	2.15e-06	4.21e-05	7.37e-05	1.76e-10
f_4	0	9,34E-04	2.67e-07	0.0015	0.0121	6.19e-07
f_5	0	3.68e-06	1.74e-08	17.06	0.0000	1.02e-08
f_6	0	5.85e-09	8.03e-09	1.60e-05	5.77e-05	7.68e-10
f_7	0	4.13e-07	1.11e-07	9.43e-04		1.00e-06
f_8	-12569,5	-12569	5.68e-09	-8,05E+04	-10400	-12600
f_9	0	8.54e-07	9.76e-08	22.88	1.28e-04	7.79e-08
f_{10}	0	6.18e-04	3.80e-08	0.0030	0.0119	1.98e-07
f_{11}	0	1.96e-04	5.84e-09	8.74e-06	0.0175	1.34e-07
f_{12}	0	2.25e-08	3.74e-09	0.1037	0.0269	1.01e-10
f_{13}	0	5.73e-09	3.76e-08	7.98e-06	0.0045	9.55e-08
f_{14}	1	0.99	1.35e-07	1.992	0.9980	0.9980
f_{15}	0.00030	0.0017	1.70e-07	3.07e-04	0.0055	6.44e-04
f_{16}	-1.0316	4.29e-08	1.22e-07	-1.03	-9,97e-01	-1,02
f_{17}	0.398	0.84	9.76e-06	0.3979	0.3984	17.17
f_{18}	3	32.68	2.53e-08	3.00	5.98	31.27
f_{19}	-3.86	-1,8997	3.16e-10	-3,86	-1,89	-1,8997
f_{20}	-3.32	-1,1698	1.08e-08	1.97e-05	-1,16	-1,1698
f_{21}	-10.1532	-10,15	1.38e-09	0.0190	-10,15	-10,1532
f_{22}	-10.4028	-10,4	5.49e-09	5.15e-05	-10,4	-10,4028
f_{23}	-10.5363	-10,53	4.22e-08	0.0017	-10,53	-10,5363

Os algoritmos aqui comparados têm característica de os agentes computacionais não convergirem para um único ponto ótimo, e as varias iterações asseguram que eles encontrem as possíveis melhores soluções para o problema a qual foram designados a

resolver. Cada algoritmo tem suas características e particularidades os diferenciando e assegurando que eles encontrem a melhor solução.

O FPA permite a busca por longas distâncias através dos polinizadores, ou seja, o algoritmo não fica preso ao mínimo local, e a lealdade a uma flor permite que o algoritmo tenha convergência mais rápida por usar sua memória. Tal algoritmo se destacados demais quando uma população de tamanho pequeno é usada, como a aqui utilizada $N = 20$.

Já no SBA, os agentes computacionais não se comunicam, diminui então o tempo de convergência do algoritmo, ele encontra o mínimo local mais rápido em comparação aos outros algoritmos. O algoritmo SBA mostrou o melhor desempenho com uma população de tamanho médio.

O algoritmo IWO tem o diferencial de dar oportunidade de reprodução a toda a população inicial, o que garante a diversidade da próxima população, e o processo de dispersão espacial diminui a possibilidade das sementes serem dispersas longe de suas "mães" o que ajusta a população. Este algoritmo se destaca com uma população de tamanho médio ou grande.

A função f_8 é uma das mais difíceis de achar o mínimo global. Para uma dimensão de 30, o mínimo desta função é aproximadamente -12569,5. Observando as tabelas com os resultados, nota-se que o algoritmo FPA possui um desempenho superior aos demais. Outra função difícil de ser solucionada é a f_5. Porém, para esta função, o algoritmo de melhor desempenho foi o PLC. No caso da função f_9 a qual apresenta uma grande dificuldade em achar o mínimo global, o algoritmo IWO apresentou melhores resultados comparado aos demais algoritmos.

Além dos resultados apresentados nas Tabelas 4.4, 4.5 e 4.6, também foram analisados os dados de consumo de memória do computador usado pelo programa MATLAB, o qual foi utilizado para executar o código do método desenvolvido, o máximo de memória utilizada pelo programa para cada matriz e o tempo de CPU, em cada função testada para cada algoritmo apresentado neste trabalho.

As Tabelas 4.7, 4.8 e 4.9 mostram os resultados de consumo de memória utilizado para executar cada matriz do algoritmo para a população específica.

Tabela 4.7 Informações de consumo de memória máxima para População=20

Funções	Memória máxima* em MB				
	FPA	SBA	IWO	PLC	AG
f_1	926	921	941	901	976
f_2	925	919	942	889	978
f_3	922	918	942	889	984
f_4	921	917	945	892	983
f_5	919	930	950	893	983
f_6	918	920	968	884	983
f_7	917	934	969	896	982
f_8	901	951	968	895	981
f_9	903	971	966	893	981
f_{10}	903	970	961	900	983
f_{11}	901	968	966	902	982
f_{12}	900	969	963	903	984
f_{13}	896	971	964	902	983
f_{14}	895	971	959	922	949
f_{15}	893	968	961	921	949
f_{16}	891	900	960	919	948
f_{17}	891	896	961	918	951
f_{18}	981	895	959	917	950
f_{19}	890	893	958	921	948
f_{20}	890	900	965	920	949
f_{21}	934	901	968	922	988
f_{22}	934	901	968	923	986
f_{23}	934	901	967	925	984

* Limitado pela memória do sistema (físico + arquivo de troca) disponível.

Tabela 4.8 Informações de consumo de memória máxima para População=50

Funções	Memória máxima* em MB				
	FPA	SBA	IWO	PLC	AG
f_1	940	930	887	898	979
f_2	940	930	887	900	972
f_3	939	935	889	904	970
f_4	938	934	904	906	967
f_5	938	938	906	906	953
f_6	939	940	901	907	951
f_7	855	940	901	908	949
f_8	915	939	903	911	939
f_9	914	938	905	912	946
f_{10}	905	938	904	915	944
f_{11}	906	936	905	915	943
f_{12}	900	941	905	919	942
f_{13}	907	941	910	924	943
f_{14}	951	943	908	918	944
f_{15}	951	942	909	920	926
f_{16}	943	948	911	921	926
f_{17}	945	949	913	924	926
f_{18}	942	949	914	924	925
f_{19}	936	948	912	927	924
f_{20}	938	947	915	928	924
f_{21}	979	950	915	926	969
f_{22}	978	951	915	924	969
f_{23}	977	950	916	924	968

* Limitado pela memória do sistema (físico + arquivo de troca) disponível.

Tabela 4.9 Informações de consumo de memória máxima para População=100

Funções	Memória máxima* em MB				
	FPA	SBA	IWO	PLC	AG
f_1	926	915	907	904	951
f_2	926	914	908	907	952
f_3	926	918	910	908	957
f_4	925	916	915	908	954
f_5	924	919	914	909	960
f_6	924	926	918	910	959
f_7	924	920	916	914	959
f_8	927	921	919	918	970
f_9	928	919	926	916	968
f_{10}	926	922	920	919	969
f_{11}	928	924	926	926	971
f_{12}	931	924	926	920	971
f_{13}	932	927	926	921	973
f_{14}	934	928	925	920	974
f_{15}	935	926	924	925	972
f_{16}	933	930	924	926	977
f_{17}	934	929	924	926	975
f_{18}	936	934	927	926	978
f_{19}	940	935	928	925	976
f_{20}	942	933	926	924	976
f_{21}	942	934	928	928	978
f_{22}	944	936	931	929	978
f_{23}	945	937	932	928	979

* Limitado pela memória do sistema (físico + arquivo de troca) disponível.

As Tabelas 4.10, 4.11 e 4.12 mostram os resultados de consumo de memória utilizado pelo programa MATLAB para executar cada função em cada algoritmo com três tamanhos de população específico.

Tabela 4.10 Informações de consumo de memória MATLAB para População=20

Funções	Memória MATLAB em MB				
	FPA	SBA	IWO	PLC	AG
f_1	5072	4813	4808	4813	4958
f_2	5074	4812	4805	4814	4820
f_3	5154	4823	4813	4815	4493
f_4	4891	4934	4815	4817	5166
f_5	4925	4935	4817	4816	5177
f_6	4869	4949	5072	4817	5191
f_7	4807	4949	5074	4817	5154
f_8	4976	4806	5154	4813	5175
f_9	4934	5072	4891	4812	5191
f_{10}	4935	5074	4925	4823	5181
f_{11}	4949	5154	4817	4780	5185
f_{12}	4949	4891	4817	5072	5177
f_{13}	4806	4925	4813	5074	5173
f_{14}	4849	5072	4812	5154	5564
f_{15}	4620	4817	4823	4891	5477
f_{16}	4810	4817	4780	4925	5837
f_{17}	4815	4813	4815	4934	5839
f_{18}	4817	4812	4817	4935	5844
f_{19}	4817	4823	4934	4949	5837
f_{20}	4813	4780	4935	4949	5839
f_{21}	4812	4958	4949	4806	5819
f_{22}	4823	4820	4949	4815	5744
f_{23}	4780	4493	4806	4817	5611

Tabela 4.11 Informações de consumo de memória MATLAB para População=50

Funções	Memória MATLAB em MB				
	FPA	SBA	IWO	PLC	AG
f_1	4789	4435	4599	4680	5617
f_2	4810	4407	4622	4435	5281
f_3	4557	4307	4455	4407	5567
f_4	5019	4172	4568	5072	5471
f_5	5030	4168	4435	5074	5927
f_6	4977	4169	4407	5154	5700
f_7	6461	5265	4307	5100	5643
f_8	5702	5072	4172	5564	5493
f_9	5593	5074	4168	5477	5715
f_{10}	5936	5154	4169	5837	5747
f_{11}	6128	5810	5178	5839	5629
f_{12}	6154	5841	5072	5844	5622
f_{13}	6141	5489	5074	5775	5644
f_{14}	5978	5564	5154	5945	5628
f_{15}	5608	5477	5839	5523	5645
f_{16}	5674	5837	5844	5517	5641
f_{17}	5667	5839	5523	5430	5634
f_{18}	5523	5844	5517	5660	5658
f_{19}	5517	5523	5430	5590	5622
f_{20}	5430	5517	5660	5955	5621
f_{21}	5660	5430	5590	5072	5710
f_{22}	5590	5660	5564	5074	5720
f_{23}	6095	5590	5587	5154	5700

Tabela 4.12 Informações de consumo de memória MATLAB para População=100

Funções	Memória MATLAB em MB				
	FPA	SBA	IWO	PLC	AG
f_1	5523	5120	5314	5118	5641
f_2	5517	5117	5304	5120	5634
f_3	5430	5125	5523	5117	5658
f_4	5641	5045	5517	5125	5622
f_5	5634	5416	5430	5045	5621
f_6	5658	5641	5660	5523	5710
f_7	5622	5634	5590	5517	5720
f_8	5621	5658	5641	5430	5700
f_9	5710	5622	5634	5660	5523
f_{10}	5720	5621	5658	5641	5517
f_{11}	5700	5710	5622	5634	5430
f_{12}	5837	5720	5621	5658	5660
f_{13}	5839	5523	5710	5622	5590
f_{14}	5844	5517	5523	5621	5837
f_{15}	5775	5837	5517	5710	5839
f_{16}	5945	5839	5837	5720	5844
f_{17}	5523	5844	5839	5523	5775
f_{18}	5517	5775	5844	5517	5945
f_{19}	5523	5945	5775	5430	5523
f_{20}	5517	5523	5945	5660	5314
f_{21}	5314	5517	5523	5590	5304
f_{22}	5304	5430	5517	5710	5293
f_{23}	5293	5660	5837	5720	5299

As Tabelas 4.13, 4.14 e 4.15 mostram os resultados de tempo de CPU em segundos, em cada função testada para cada algoritmo apresentado nesse trabalho.

Tabela 4.13 Informações de consumo de CPU para População=20

Funções	Tempo de CPU (segundos)				
	FPA	SBA	IWO	PLC	AG
f_1	4412	1091	1799	1799	3765
f_2	4521	1222	2080	2080	3557
f_3	4809	1433	2395	2395	4167
f_4	4908	1799	1091	2741	4525
f_5	5046	2080	1222	2978	4752
f_6	5145	2395	1433	3059	5005
f_7	5254	2741	1799	3165	5409
f_8	5341	2894	2080	3765	5908
f_9	5424	2678	2395	3557	6057
f_{10}	5512	2984	2497	4167	6376
f_{11}	5590	2999	3765	4525	6602
f_{12}	5648	2978	3557	4752	6631
f_{13}	5747	3059	4167	4412	6686
f_{14}	5943	3165	4525	4521	5995
f_{15}	5994	3256	4412	4809	8473
f_{16}	6005	3487	4521	4908	3965
f_{17}	6045	3384	4809	5046	1091
f_{18}	6085	3897	4908	5145	1222
f_{19}	6144	3765	5046	3765	1433
f_{20}	6206	3557	5145	3557	1799
f_{21}	6276	4167	5697	4167	2080
f_{22}	6358	4525	5884	4525	2395
f_{23}	6458	4987	5796	4698	2741

Tabela 4.14 Informações de consumo de CPU para População=50

Funções	Tempo de CPU (segundos)				
	FPA	SBA	IWO	PLC	AG
f_1	6786	4192	4178	4099	4278
f_2	7120	4355	4184	4192	4551
f_3	7790	4471	4199	4355	5921
f_4	8128	4603	4192	4471	5087
f_5	8479	4720	4355	4603	5768
f_6	8797	4722	4471	4720	5239
f_7	8364	4723	4603	4722	5168
f_8	7745	4725	4720	4723	5641
f_9	7137	5087	4722	4725	5872
f_{10}	7532	5768	4723	5087	5941
f_{11}	7190	5239	4725	5768	6467
f_{12}	7174	5168	5087	5444	6488
f_{13}	7596	5641	5768	6422	6591
f_{14}	7530	5444	5239	6560	6088
f_{15}	7696	6422	5168	6689	6310
f_{16}	7109	6560	5641	6794	7434
f_{17}	6831	6689	5872	6906	7555
f_{18}	6963	6794	5444	6591	7683
f_{19}	7155	6906	6422	6088	7890
f_{20}	7135	7093	6560	6310	7237
f_{21}	7158	7237	6689	7434	7530
f_{22}	7829	7530	6794	7555	7844
f_{23}	7211	7844	6906	7683	8191

Tabela 4.15 Informações de consumo de CPU para População=100

Funções	Tempo de CPU (segundos)				
	FPA	SBA	IWO	PLC	AG
f_1	6422	6264	5697	5757	7434
f_2	6560	6874	5884	5784	7555
f_3	6689	6422	5796	5796	7683
f_4	6794	6560	6257	5697	7890
f_5	6906	6689	6422	5884	7237
f_6	7093	6794	6560	5796	7530
f_7	7237	6906	6689	6422	7844
f_8	7530	7093	6794	6560	8191
f_9	7844	7237	6906	6689	7434
f_{10}	7689	7530	7093	6794	7555
f_{11}	7434	7844	7237	6906	7683
f_{12}	7555	7987	7530	7093	7205
f_{13}	7683	7648	7844	7237	7368
f_{14}	7890	7434	7093	7530	7120
f_{15}	7237	7555	7279	7844	7790
f_{16}	7530	7683	7528	6422	8128
f_{17}	7844	7890	7093	7364	8479
f_{18}	7137	7237	7434	7434	8797
f_{19}	7532	7530	7555	7555	8364
f_{20}	7190	7555	7683	7683	7120
f_{21}	7174	7683	7890	7890	7790
f_{22}	7596	7205	7237	7237	8128
f_{23}	7530	7258	7159	7530	8479

Os resultados apresentados indicam que não existe um algoritmo melhor do que outro quando considera os infinitos problemas existentes, que está de acordo com o teorema da inexistência de almoço grátis (*No Free Lunch* - NFL). Os algoritmos desenvolvidos ao longo desta pesquisa, quando comparados, apresentam bons resultados para um determinado tipo de função, por exemplo, para a função f_5 o que apresentou melhores resultados foi o PLC, para a função f_8 foi o FPA, e assim por diante. Então, é interessante o emprego destes algoritmos na resolução de práticos problemas encontrados em diversas áreas existentes.

Capítulo 5

5. Modificação do algoritmo IWO

Neste capítulo será apresentada uma proposta de modificação para o algoritmo de colonização das ervas daninhas (*Invasive Weed Optimization* - IWO). A seção 5.1 traz uma introdução ao capítulo. O algoritmo IWO-M é apresentado na seção 5.2. A seção 5.3 mostra a performance do algoritmo *Invasive Weed Optimization Modified* (IWO-M).

5.1. Introdução

O algoritmo IWO-M proposto é baseado na colonização de plantas daninhas, as plantas daninhas são qualquer tipo de planta invasora do espaço ao qual foi cultivada a planta desejada. O algoritmo tem como objetivo garantir a diversidade de sua população do inicio ao fim. Ele é uma modificação do algoritmo bioinspirado IWO.

5.2. IWO-M

Baseado no processo ecológico de ervas invasoras, o IWO é um algoritmo de otimização capaz de resolver problemas multidimensionais, lineares e não lineares, já utilizado para solução de diversos problemas de engenharia, como apresentado na seção 3.4. Tendo a elaboração do algoritmo considerando as plantas com flores, sua simulação se baseia em três principais etapas do ciclo de vida de uma planta, são elas: reprodução, dispersão de sementes e exclusão competitiva.

- A etapa de reprodução garante que todos os indivíduos, independente de sua aptidão, reproduza. A produção de sementes acontece de forma linear, ou seja, o indivíduo com melhor aptidão produz mais sementes e o com menor aptidão produz menos sementes;
- A etapa de dispersão espacial de sementes segue as diretrizes da distribuição normal, assegurando que as sementes produzidas sejam dispersas perto da erva mãe;
- Na etapa de exclusão competitiva, as plantas com melhor aptidão são selecionas para a nova a população e as de pior aptidão são excluídas.

Essa técnica de reprodução proposta dá uma chance para indivíduos inviáveis sobreviver e reproduzir semelhante ao mecanismo que acontece na natureza, pois acredita-se na possibilidade de alguns dos indivíduos inviáveis transporte informação mais útil do que os indivíduos viáveis durante o processo de evolução [31].

Já a técnica de exclusão competitiva garante que apenas as plantas com melhor aptidão sejam selecionadas a formar a nova população, o que de certa forma contradiz a técnica de reprodução anteriormente realizada.

Assim como o IWO, o algoritmo IWO-M segue também as três principais etapas do ciclo de vida de uma planta florida: reprodução, dispersão de sementes e exclusão competitiva.

A alteração proposta é na etapa de exclusão competitiva, que passa a ser seleção por roleta, pois acredita que a exclusão competitiva adotada pelo IWO não garante a diversidade do algoritmo, não dando oportunidade aos indivíduos inferiores (com aptidão ruim), já que os mesmo são excluídos, sobrevivendo apenas os melhores.

Na etapa de seleção por roleta, a cada indivíduo da população é atribuída uma fatia de uma circular roleta, o tamanho da fatia é proporcional à aptidão do indivíduo. A roda é girada N vezes, onde N é o de indivíduos na população. Em cada rotação, o indivíduo sob o marcador da roda é selecionado para estar entre os indivíduos da próxima geração [4]. Este método pode ser implementado de acordo com a Figura 5.1.

O fluxograma representando o algoritmo IWO-M é apresentado na Figura 5.2.

Algoritmo seleção por roleta
$A \leftarrow \sum_{i=1}^{n} f_i$ soma das aptidões de todos os indivíduos da população total_parcial← 0 **Enquanto** o critério de parada não for satisfeito **faça** $rand \leftarrow$ valor aleatório $[0, A]$ $total\_parcial \leftarrow total\_parcial + f_i$ **Se** $total\_parcial >= rand$ Selecione o indivíduo corrente **Fim se** **Fim enquanto**

Figura 5.1 Algoritmo básico do método de seleção por roleta [7].

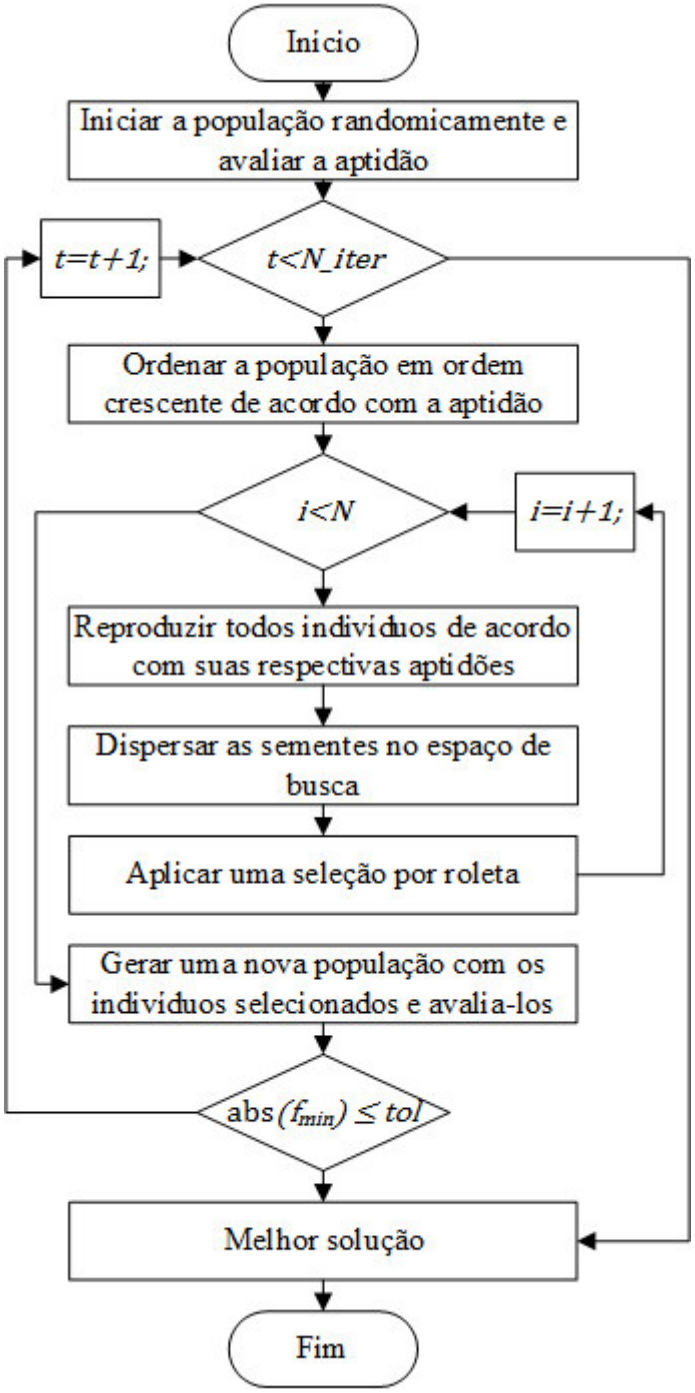

Figura 5.2 Fluxograma algoritmo IWO-M.

5.3. Performance do IWO-M

O algoritmo IWO-M foi testado com as funções benchmarks apresentadas no Capítulo 4. Utilizando parâmetros iniciais definidos como: o número de iterações (*N_iter)* 10000, tolerância (*tol*) como condição de término igual a 1e-6. O algoritmo foi testado para três tamanhos de população (*N*) igual a 20, 50 e 100, sendo que cada função foi

testada 10 vezes, e tirado seu mínimo, média e desvio padrão, como mostram as Tabelas 5.1, 5.2 e 5.3.

Tabela 5.1 Resultados algoritmo IWO-M para População = 20

Funções	f_{min}	Mínimo	Média	Desvio Padrão
f_1	0	2.53e-07	3.44e-06	0.000002
f_2	0	3.24e-06	3.24e-06	0.000002
f_3	0	1.91e-07	4.19e-06	0.000003
f_4	0	5.79e-07	4.64e-06	0.000003
f_5	0	28.96	28.99	0.015412
f_6	0	0.0000	0.0000	0.000000
f_7	0	8.32e-08	4.58e-06	0.000004
f_8	-12569,5	-2.64e+03	-2.07e+03	451.49
f_9	0	2.28e-10	4.13e-06	0.000004
f_{10}	0	1.65e-06	5.70e-06	0.000002
f_{11}	0	3.59e-07	3.58e-06	0.000002
f_{12}	0	1.66	1.66	0.001775
f_{13}	0	4.79e-07	2.09	1.37
f_{14}	1	12.67	12.67	0.000000
f_{15}	0.00030	0.0035	0.0741	0.048493
f_{16}	-1.0316	7.47e-06	1.57e-06	0.000003
f_{17}	0.398	55.57	55.59	0.011229
f_{18}	3	600	600	0.000000
f_{19}	-3.86	0.0694	-0,0684	0.000536
f_{20}	-3.32	-0.0054	-0.0051	0.000083
f_{21}	-10.1532	-0.2736	-0.2736	0.000136
f_{22}	-10.4028	-0.2936	-0.2936	0.000009
f_{23}	-10.5363	-0.3222	-0.3218	0.000158

Tabela 5.2 Resultados algoritmo IWO-M para População = 50

Funções	f_{min}	Mínimo	Média	Desvio Padrão
f_1	0	1.48e-07	3.78e-06	0.000002
f_2	0	6.05e-07	4.89e-06	0.000002
f_3	0	1.19e-07	4.03e-06	0.000003
f_4	0	5.81e-07	4.56e-06	0.000003
f_5	0	29	29	0.015412
f_6	0	7.49	7.49	0.000000
f_7	0	4.30e-06	0.0594	0.000004
f_8	-12569,5	-3.27e+03	-2.36e+03	451.49
f_9	0	2.63e-09	1.85e-06	0.000004
f_{10}	0	9.46e-08	5.29e-06	0.000002
f_{11}	0	2.77e-08	2.78e-06	0.000002
f_{12}	0	0.9828	1.58	0.001775
f_{13}	0	3	3	1.37
f_{14}	1	3.17	8.90	0.000000
f_{15}	0.00030	0.0027	0.0529	0.048493
f_{16}	-1.0316	-1.0205	-3.97e-01	0.000003
f_{17}	0.398	0.4211	0.9849	0.011229
f_{18}	3	3.19	20.33	0.000000
f_{19}	-3.86	-3.7702	-3.5328	0.000536
f_{20}	-3.32	-2.3075	-1.5111	0.000083
f_{21}	-10.1532	-3.4682	-1.91	0.000136
f_{22}	-10.4028	-3.8349	-2.0844	0.000009
f_{23}	-10.5363	-3.5579	-1.807	0.000158

Tabela 5.3 Resultados algoritmo IWO-M para População = 100

Funções	f_{min}	Mínimo	Média	Desvio Padrão
f_1	0	1.26e-06	4.88e-06	0.000003
f_2	0	2.47e-06	6.29e-06	0.000002
f_3	0	3.13e-09	4.38e-06	0.000003
f_4	0	1.04e-06	5.26e-06	0.000003
f_5	0	29	29	0.000000
f_6	0	7.27	7.47	0.066879
f_7	0	1.27e-06	0.3610	0.307808
f_8	-12569,5	-3.50e+03	-2.41e+03	419.18
f_9	0	3.30e-10	2.79e-06	0.000004
f_{10}	0	1.94e-07	4.73e-06	0.000002
f_{11}	0	1.52e-07	3.80e-06	0.000003
f_{12}	0	1.58	1.65	0.025001
f_{13}	0	3.00	3.00	0.000000
f_{14}	1	2.53	9.52	3.483
f_{15}	0.00030	0.0066	0.0288	0.014689
f_{16}	-1.0316	-1.01	-5.30e-01	0.431619
f_{17}	0.398	0.4382	0.9200	0.494791
f_{18}	3	3.06	15.02	9.747
f_{19}	-3.86	-3.81	-3.63	0.139599
f_{20}	-3.32	-2.92	-2.14	0.426605
f_{21}	-10.1532	-4.64	-2.25	1.413
f_{22}	-10.4028	-4.09	-2.21	1.017
f_{23}	-10.5363	-4.18	-2.50	1.264847

Em geral o algoritmo proposto é relativamente bom, com destaque para função f_9, a qual apresenta uma grande dificuldade em achar o mínimo global, o algoritmo IWO-M apresentou melhores resultados comparado ao algoritmo IWO para os três tamanhos de população.

Além dos resultados de apresentados nas Tabelas 5.1, 5.2 e 5.3, também foram analisados os dados de consumo de memória do computador usado pelo programa

MATLAB, o qual foi utilizado para executar o código do método desenvolvido, o número máximo de memória utilizada pelo programa para cada matriz e o tempo de CPU, em cada função testada. A Tabela 5.4 mostra os resultados encontrados.

Tabela 5.4 Informações de consumo do algoritmo IWO-M para População=20

Funções	Memória (MB)		CPU
	Máxima *	MATLAB	
f_1	4802	958	772
f_2	4806	957	774
f_3	4799	963	776
f_4	4804	963	777
f_5	4807	962	847
f_6	4802	960	937
f_7	4807	959	991
f_8	4809	959	1037
f_9	4811	957	1039
f_{10}	4813	957	1041
f_{11}	4814	957	1043
f_{12}	4820	948	1178
f_{13}	4817	947	1317
f_{14}	4823	945	2369
f_{15}	4805	944	2424
f_{16}	4817	943	2438
f_{17}	4820	941	2496
f_{18}	4818	939	2539
f_{19}	4825	939	2613
f_{20}	4811	937	2688
f_{21}	4818	980	2789
f_{22}	4821	981	2908
f_{23}	4828	979	3058

* Limitado pela memória do sistema (físico + arquivo de troca) disponível.

Tabela 5.5 Informações de consumo do algoritmo IWO-M para População=50

Funções	Memória (MB)		CPU
	Máxima *	MATLAB	
f_1	4600	1077	4170
f_2	4610	1077	4170
f_3	4604	1078	4180
f_4	4609	1075	4192
f_5	4680	1074	4355
f_6	4435	1073	4471
f_7	4407	1072	4603
f_8	4307	1072	4720
f_9	4172	1072	4722
f_{10}	4168	1073	4723
f_{11}	4169	1073	4725
f_{12}	5178	1075	5117
f_{13}	5270	1074	5444
f_{14}	4734	1073	6422
f_{15}	5101	1074	6560
f_{16}	5117	1073	6689
f_{17}	5100	1073	6794
f_{18}	5089	1072	6906
f_{19}	5104	1072	7093
f_{20}	5119	1072	7279
f_{21}	5125	1071	7528
f_{22}	4296	1072	8162
f_{23}	4079	1073	8602

* Limitado pela memória do sistema (físico + arquivo de troca) disponível.

Tabela 5.6 Informações de consumo do algoritmo IWO-M para População=100

Funções	Memória (MB)		CPU
	Máxima *	MATLAB	
f_1	5467	942	840
f_2	5469	943	842
f_3	5484	941	847
f_4	5483	948	849
f_5	5488	936	1178
f_6	5449	937	1401
f_7	5444	935	1708
f_8	5253	930	2150
f_9	5120	931	2152
f_{10}	5117	939	2154
f_{11}	5125	933	2155
f_{12}	5045	929	2820
f_{13}	5804	926	3454
f_{14}	5295	923	5322
f_{15}	5287	920	5606
f_{16}	5294	920	5771
f_{17}	5311	905	5981
f_{18}	5306	907	6199
f_{19}	5302	908	6577
f_{20}	5314	906	6963
f_{21}	5304	898	7462
f_{22}	5293	896	8054
f_{23}	5694	946	8372

* Limitado pela memória do sistema (físico + arquivo de troca) disponível.

O algoritmo IWO-M diferencia-se dos demais algoritmos por permitir que todos os indivíduos tenham oportunidade de sobreviver e reproduzir, não apenas na etapa de reprodução, mas também na etapa de seleção, o que garante a diversidade do algoritmo.

Capítulo 6

6. Conclusão

Neste capítulo são apresentadas as conclusões do trabalho e as sugestões de novos tópicos a serem pesquisados em trabalhos futuros.

6.1. Conclusões

- Os algoritmos FPA, SBA, IWO e PLC, baseados na inteligência das plantas, foram simulados para a otimização das funções de teste. Os resultados demonstraram que este algoritmos são eficientes, porém apresentam uma desagradável característica dos algoritmos estocásticos: são imprevisíveis. Ou seja, supondo que execute os algoritmos uma vez e encontre o mínimo da função objetivo *f*. Se for executado novamente estes algoritmos, provavelmente irá encontrar uma solução diferente de *f*. Um bom índice de desempenho para compara a eficiência de algoritmos de IA é calculando a média e o desvio padrão das soluções quando forem executados N vezes. Quanto menor o valor de desvio padrão mais preciso é o algoritmo, pois o desvio padrão de uma amostra dar-nos uma medida de dispersão;
- Quando executados N vezes estes algoritmos apresentaram bons resultados, porém há casos onde as soluções não foram precisas, não foram exatas e, as vezes, nem precisas e nem exatas. Precisão é diferente de exatidão. A precisão indica o quanto as medidas repetidas estão próximas umas das outras, em contrapartida, o grau de exatidão assinala o quão próximo do valor real (no caso dos problemas de otimização o valor real é considerado a solução do problema), está o valor calculado pelo algoritmo. Resumidamente, o algoritmos de IA para ser considerado "bom" será necessário precisão e exatidão, ademais deve-se considerar outros índices como o tempo de execução e o número de iterações para atingir a solução levando em conta a tolerância;
- Cada algoritmo tem suas características e particularidade, os diferenciando pela maneira que a melhor solução é encontrada. Ademais, estes algoritmos

apresentam poucos parâmetros de entrada, o que torna uma característica positiva quando comparados com os clássicos algoritmos de IA, tais como o PSO e o GA;

- O algoritmo FPA permite a pesquisa em longas distâncias através de polinizadores, ou seja, o algoritmo não está vinculado ao mínimo local. A lealdade a uma flor permite ao algoritmo a convergência mais rápida;
- No algoritmo SBA os agentes computacionais não se comunicam, então diminui o tempo de convergência do algoritmo, ele atende o mínimo mais rápido em comparação com outros algoritmos;
- O algoritmo IWO tem a característica de dar a oportunidade a todos os indivíduos da população serem reproduzidos (mais aptos e menos aptos a encontrarem o ótimo), aumentando a população rapidamente. Além de demonstrar características interessantes para escapar dos mínimos locais.
- O algoritmo PLC mostrou também sua eficiência quando teve o melhor desempenho com uma das funções mais difíceis de ser solucionada, a função f_5;
- Os resultados apresentados indicam que não existe um algoritmo melhor do que outro quando considera os infinitos problemas existentes, cada algoritmo tem suas particularidades, garantindo que a melhor solução seja encontrada e que seja solução seja a ideal para o problema a qual foi empregado. Todos os algoritmos convergiram para um mínimo, escaparam de mínimos locais do seu jeito, e cada um se destacou em alguma função.

6.2. Trabalhos futuros

Existem várias possibilidades de futuros desenvolvimentos relacionados com este trabalho de pesquisa.

- Aperfeiçoamento do algoritmo IWO-M;
- Aplicações deste algoritmos a problemas práticos de otimização combinatória;
- Criações de novos algoritmos baseados na inteligência das plantas;

Referências

[1] S. Russell, P. Norvig, *Inteligência Artificial*, 3ª ed., v.1, Elsevier Brasil, 2014.

[2] G. F. Luger, *Inteligência Artificial*, 6ª ed., Pearson, 2013.

[3] A. P. Engelbrecht, *Computational Intelligence: an introduction*, John Wiley & Sons, 2nd ed., 2007.

[4] M. Mitchell, *An Introduction to Genetic Algorithms*, The MIT Press, 1999.

[5] E. G. M. de Lacerda, A. C. P. L. F. de Carvalho, "Introdução aos algoritmos genéticos," *Sistemas inteligentes: aplicações a recursos hídricos e ciências ambientais*, v. 1, p. 99-148, 1999.

[6] A. B. S. Serapião, "Fundamentos de otimização por inteligência de enxames: uma visão geral," *SBA: Controle & Automação Sociedade Brasileira de Automática*, v. 20, n. 3, p. 271-304, 2009.

[7] M. Dorigo, L. M. Gambardella, "Ant Colony System: A Cooperative Learning," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53-66, Apr., 1997.

[8] A. Carvalho, A. Delbem, R. Romero, E. Simões, G. Telles, "Computação Bioinspirada," Em: SBC. (Org.). Congresso da Sociedade Brasileira de Computação, 24.; Jornada de Atualização em Informática. v. 2, p. 145-191, 2004.

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

[10] E. D. Brenner, R. Stahlberg, S. Mancuso, J. Vivanco, F. Baluska, E. V. Volkenburgh, "Plant neurobiology: an integrated view of plant signaling," *Trends in Plant Science*, vol. 11, pp. 413-419, Aug. 2006.

[11] M. Pollan, *The Intelligent Plant*, The New Yorker, 2013.

[12] M. Sulaiman, A. Salhi, and E. S. Fraga, "The plant propagation algorithm: modifications and implementation," arXiv preprint arXiv:1412.4290, vol. 1, pp. 1–10, Dec 2014.

[13] J. M. McNamara and A. I. Houston, “The plant propagation algorithm: modifications and implementation,” *Nature*, vol. 380, no. 6571, pp. 215– 221, Mar 1996.

[14] C. D. Schlichting and M. Pigliucci, *Phenotypic Evolution: A Reaction Norm perspective*. Sinauer Associates Incorporated, 1998.

[15] M. Karami, A. Moosavinia, M. Ehsanian, and M. Teshnelab, “A new evolutionary optimization algorithm inspired by plant life cycle,” in *Proc. of 23rd Iranian Conference on Electrical Engineering*, Tehran, Iran, 2015, pp. 573–577.

[16] A. Trewavas, “Green plants as intelligent organisms,” *Trends in Plant Science*, vol. 10, no. 9, pp. 413-419, Sep. 2005.

[17] A. Trewavas, “Aspects of plant intelligence,” *Annals of Botany*, vol. 92, no. 1, pp. 1–20, May 2003.

[18] A. Trewavas, “Plant Intelligence,” *Naturwissenschaften*, vol. 92, pp. 401-413, 2005.

[19] A. Singh, “Plant intelligence: A new approach to soft computing,” in *Proc. of 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, Haryana, India, 2015, pp. 1429– 1432.

[20] A. Singh, “A review of intelligent attributes in plants,” in *Proc. of International Multi Conference on Intelligent Systems and Nanotechnology (IISN)*, Haryana, India, 2010, pp. 26–28.

[21] X.-S. Yang, “Flower pollination algorithm for global optimization,” in *Proc. of Int. Conference on Unconventional Computing and Natural Computation*, Berlin, Heidelberg, pp. 240–249, 2012.

[22] L. Kanagasabai, B. RavindhranathReddy, “Reduction of real power loss by using fusion of flower pollination algorithm with particle swarm optimization,” *Journal of the Institute of Industrial Applications Engineers*, vol. 2, no.3, pp.97–103, 2014.

[23] R. Prathiba, M. B. Moses, S. Sakthivel, “Flower pollination algorithm applied for different economic load dispatch problems,” *Int. Journal of Engineering and Technology*, vol.6, no.2, pp.1009–1016, 2014.

[24] O. A. Raouf, M. A. Baset, I. Elhenawy, “A novel hybrid flower pollination algorithm with chaotic harmony search for solving sudoku puzzles,” *Int. Journal Modern Education and Computer Science*, v. 3, pp.33-44, 2014.

[25] G. M. Platt, “Computational Experiments with Flower Pollination Algorithm in the Calculation of Double Retrograde Dew Points,” *Int. Review of Chemical Engineering*, vol.6, no.2, pp. 95-99, 2014.

[26] O. A. Raouf, M. A. Baset, I. Elhenawy, “A new hybrid flower pollination algorithm for solving constrained global optimization problems,” *Int. Journal of Applied Operational Research*, vol.4, no.2, pp. 1–13, 2014.

[27] S. Akyol, B. Alatas, “Plant intelligence based metaheuristic optimization algorithms,” *Artificial Intelligence Review*, pp. 1-46, 2016.

[28] B. J. Glover, *Understanding flowers and flowering: an integrated approach*. UK: Oxford University Press, 2007, vol. 277.

[29] M. Amaya-Márquez, “Floral constancy in bees: a revision of theories and a comparison with other pollinators,” *Revista Colombiana de Entomología*, vol. 35, no. 2, pp. 206–216, 2009.

[30] I. Pavlyukevich, “Lévy flights, non-local search and simulated annealing,” *Journal of Computational Physics*, vol. 226, no. 2, pp. 1830–1844, Oct. 2007.

[31] A. R. Mehabian and C. Lucas, “A novel numerical optimization algorithm inspired from weed colonization,” in *Ecological Informatics*, v. 1, n. 4, p. 355–366, 2006.

[32] B. Dadalipour, A. R. Mallahzadeh, Z. Davoodi-Rad, “Application of the invasive weed optimization technique for antenna configurations,” *Antennas and Propagation Conference*, IEEE, pp.425-428, 2008.

[33] G. G. Roy, S. Das, P. Chakraborty, P. N. Suganthan, “Design of non-uniform circular antenna arrays using a modified invasive weed optimization algorithm,” *IEEE Trans. on Antennas and Propagation*, vol. 59, no. 1, p. 110-118, 2011.

[34] S. Karimkashi, A. A. Kishk, “Invasive weed optimization and its features in electromagnetics,” *IEEE Trans. on Antennas and Propagation*, vol. 58, no. 4, pp. 1269-1278, 2010.

[35] E. Pourjafari, H. Mojallali, “Solving nonlinear equations systems with a new approach based on invasive weed optimization algorithm and clustering,” *Swarm and Evolutionary Computation*, v. 4, p. 33-43, 2012.

[36] A. M. A. da Silva, I. D. Coelho, P. R. de Medeiros, “Levantamento florístico das plantas daninhas em um parque público de Campina Grande, Paraíba, Brasil,” *Biotemas*, v. 21, n. 4, pp.7-14, 2011.

[37] F. Merrikh-Bayat, “A Numerical Optimization Algorithm Inspired by the Strawberry Plant,” CoRR, vol. abs/1407.7399, 2014.

[38] M. Karami, A. Moosavinia, M. Ehsanian, and M. Teshnelab, “A new evolutionary optimization algorithm inspired by plant life cycle,” in *Proc. of 23rd Iranian Conference on Electrical Engineering*, Tehran, Iran, 2015, pp. 573–577.

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

[40] M. Jamil and X.-S. Yang, “A literature survey of benchmark functions for global optimisation problems,” *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.

[41] S. Mirjalili, “Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm,” *Knowledge-Based Systems*, vol. 89, pp. 228 – 249, 2015.

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

[43] C. J. Chung, R. G. Reynolds, “CAEP: An Evolution-Based Tool for Real-Valued Function Optimization Using Cultural Algorithms,” *International Journal on Artificial Intelligence Tool*, vol. 7, no. 3, pp. 239-291, 1998.

[44] P. H. Winston, *Artificial Intelligence*, 3rd ed., Addison-Wesley, 1992.